UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO DE INTEGRACIÓN CURRICULAR

TEMA:

"ARQUITECTURA MULTIAGENTE CON EL PROTOCOLO A2A PARA EXTRACCIÓN DE CLÁUSULAS EN CONTRATOS SAAS MEDIANTE LLMS OPEN-SOURCE"

Trabajo de titulación previo a la obtención del título de Ingeniero en Tecnologías de la Información

Línea de investigación: Desarrollo, aplicación de software y cyber security (seguridad cibernética)

AUTOR:

Patricio Roberto Paspuel Armas

DIRECTOR:

MSc. Pablo Andrés Landeta López

Ibarra – Ecuador 2026

UNIVERSIDAD TÉCNICA DEL NORTE
DIRECCIÓN DE BIBLIOTECA

AUTORIZACIÓN DE USO Y PUBLICACIÓN
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO	
APELLIDOS Y NOMBRES:	Paspuel Armas Patricio Roberto

DATOS DE LA OBRA	
TÍTULO:	ARQUITECTURA MULTIAGENTE CON EL PROTOCOLO A2A PARA EXTRACCIÓN DE CLÁUSULAS EN CONTRATOS SAAS MEDIANTE LLMS OPEN-SOURCE
AUTOR (ES):	Paspuel Armas Patricio Roberto
FECHA: DD/MM/AAAA	26/08/2026
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniero en Tecnologías de la Información
ASESOR /DIRECTOR:	PhD. Cathy Pamela Guevara Vega / MSc. Pablo Andrés Landeta López

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 26 días del mes de agosto de 2026

EL AUTOR:

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
PATRICIO ROBERTO
PASPUEL ARMAS

(Firma)……
Nombre: Paspuel Armas Patricio Roberto

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 26 de agosto de 2026.

MSc. Landeta López Pablo Andrés
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:
Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
PABLO ANDRES LANDETA LOPEZ

MSc. Landeta López Pablo Andrés

C.C: 1002161055

APROBACIÓN DEL COMITÉ CALIFICADOR

El Comité Calificado del trabajo de Integración Curricular “ Arquitectura multiagente con el protocolo A2A para extracción de cláusulas en contratos SaaS mediante LLMs Open-Source ” elaborado por Paspuel Armas Patricio Roberto , previo a la obtención del título del Ingeniero en Tecnologías de la Información , aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte:

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
PABLO ANDRES
LANDETA LOPEZ

MSc. Landeta López Pablo Andrés

C.C: 1002161055

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
CATHY PAMELA
GUEVARA VEGA

PhD. Guevara Vega Cathy Pamela

C.C: 1002334835

DEDICATORIA

A mis padres, pilares fundamentales de todo lo que soy. A mi madre, en especial, cuyo sacrificio silencioso, fe inquebrantable y presencia constante han sido la raíz desde la cual ha crecido cada uno de mis logros; este título lleva tu nombre grabado en cada página. A mi padre, cuya fortaleza y ejemplo me enseñaron que los grandes propósitos se construyen con constancia, trabajo y dignidad.

A mis hermanos, quienes con una palabra de aliento en el momento justo o con ese apoyo silencioso que solo quien te conoce profundamente sabe dar, me recordaron en cada etapa que no estaba solo. A mis amigos, por su lealtad y esa complicidad que solo una amistad verdadera puede ofrecer, siendo mi recordatorio más fiel de que los caminos difíciles se recorren mejor en buena compañía.

A mi director de tesis y a mi asesora, quienes más allá de ser excelentes profesionales son personas de una calidad humana extraordinaria. En cada momento de incertidumbre supieron estar presentes con la palabra exacta y la serenidad necesaria para esfumar cualquier obstáculo. Su generosidad, paciencia y vocación dejan una huella que va mucho más allá de esta investigación.

Todos ustedes son los coautores verdaderos de este trabajo; con paciencia y entrega hicieron posible que llegara hasta este punto. Este logro no me pertenece solo a mí, es de cada uno de ustedes, que con entrega, presencia y apoyo incondicional, convirtieron un sueño académico en una realidad que hoy sostengo con orgullo.

Paspuel Armas Patricio Roberto

AGRADECIMIENTO

Quisiera comenzar expresando mi más profundo y sincero agradecimiento a Dios, quien ha sido mi guía y fortaleza a lo largo de este camino académico.

A la Universidad Técnica del Norte, y en especial a la Facultad de Ingeniería en Ciencias Aplicadas y a la Carrera de Tecnologías de la Información, por brindarme el espacio, los recursos y la formación necesaria para desarrollar esta investigación.

A mi director de tesis, MSc. Pablo Landeta López, por su orientación constante, su exigencia académica y su disposición para guiarme en cada etapa de este proceso, siendo determinante para alcanzar los resultados que hoy se presentan.

A mi asesora, PhD. Cathy Guevara Vega, por compartir generosamente su experiencia y conocimiento metodológico, cuyas indicaciones oportunas contribuyeron de manera significativa a elevar la calidad de esta investigación.

A los docentes que formaron parte de mi trayectoria universitaria, tanto a quienes siguen compartiendo su conocimiento en las aulas como a aquellos que ya no están presentes pero cuyas enseñanzas permanecen. Cada uno dejó una huella en mi formación profesional y humana.

A mis compañeros y amigos de facultad, con quienes compartí largas jornadas de estudio, debates, dudas y logros, enriqueciendo mi perspectiva como investigador.

Finalmente, y con especial gratitud, agradezco a mi familia, pilar fundamental de todo lo que soy. Su paciencia, apoyo incondicional y fe en mí fueron el impulso que me motivó a continuar en los momentos más exigentes. Este logro es también de ustedes.

Paspuel Armas Patricio Roberto

ÍNDICE DE CONTENIDOS

DEDICATORIA II

AGRADECIMIENTO III

ÍNDICE DE FIGURAS VIII

ÍNDICE DE TABLAS VIII

RESUMEN X

ABSTRACT XI

CAPÍTULO I

1.1 Planteamiento del problema 1

1.2 Objetivos 2

1.2.1 Objetivo General 2

1.2.2 Objetivos Especificos 2

1.3 Alcance 3

1.4 Justificación 5

1.4.1 Beneficios 6

CAPÍTULO II

2.1 Antecedentes 8

2.2 Contratos SaaS y Análisis Jurídico 11

2.2.1 Estructura y componentes críticos de contratos SaaS 12

2.2.2 Tipología de cláusulas legales 13

2.3 Modelos de lenguaje a gran escala 14

2.3.1 Arquitecturas de LLMs open-source 15
2.3.2 Fine-tuning para dominio jurídico 18
2.3.3 Limitaciones en extracción de cláusulas 21
2.4 Desafíos en la interpretación automatizada de lenguaje jurídico 24
2.5 Sistemas Multiagente (MAS) 26
2.5.1 Fundamentos de arquitecturas multiagente 27
2.5.2 Roles y especialización de agentes 31
2.5.3 Aplicaciones en procesamiento de documentos legales 33
2.6 Protocolo A2A 34
2.6.1 Actores del Protocolo A2A 35
2.6.2 Componentes Principales de A2A 37
2.6.3 Capa de transporte A2A y Comunicación 38
2.6.4 Ciclo de vida del Agente Remoto (Servidor) 38
2.6.5 Estándares de comunicación 39
2.6.6 Mecanismos de coordinación 42
2.6.7 Interoperabilidad entre frameworks 45
2.7 Extracción Automatizada de Cláusulas 46
2.7.1 Técnicas de NLP para documentos legales 47
2.7.2 Almacenamiento y recuperación de cláusulas 49
2.7.3 Evaluación de precisión 53
2.8 Tecnologías de Implementación 54

2.8.1 Frameworks para agentes 55
2.8.2 Plataformas web 56
2.8.3 Herramientas de despliegue 57
2.9 Métricas de Evaluación para Agentes IA 59
2.9.1 Eficiencia de coordinación 59
2.9.2 Calidad de respuestas 61
2.9.3 Escalabilidad 63
CAPÍTULO III
3.1 Enfoque y Tipo de Investigación 66
3.1.1 Investigación aplicada en inteligencia artificial 66
3.1.2 Metodología de desarrollo basada en prototipado. 69
3.2 Diseño de la Arquitectura Multiagente 78
3.2.1 Especificación de agentes 79
3.2.2 Protocolo A2A para coordinación. 83
3.3 Desarrollo de la Aplicación Web 84
3.3.1 Requisitos funcionales y no funcionales 85
3.3.2 Integración de LLMs y base de datos 87
3.4 Métricas de Evaluación 90

3.4.1 Precisión en identificación de cláusulas 90
3.4.2 Eficiencia en consultas y escalabilidad 92
3.4.3 Análisis comparativo 93
CAPÍTULO IV
4.1 Especificaciones del Sistema 96
4.1.1 Diagrama de la arquitectura multiagente 96
4.1.2 Flujo de procesamiento de contratos 98
4.2 Implementación 99
4.2.1 Entrenamiento y ajuste de LLMs para lenguaje jurídico 99
4.2.2 Interfaz web para consultas de usuarios 100
4.3 Pruebas y Validación 101
4.3.1 Casos de uso con contratos SaaS reales 101
4.3.2 Comparación con métodos manuales y otras herramientas 105
4.4 Análisis de Métricas 106
4.4.1 Rendimiento de los agentes 106
4.4.2 Satisfacción de usuarios piloto 117
CONCLUSIONES 126
RECOMENDACIONES 128
BIBLIOGRAFÍA 130
ANEXOS 140

ÍNDICE DE FIGURAS

Figura 1 Análisis causal del problema SaaS 2
Figura 2 Arquitectura general 4

Figura 3 Pila de prompting legal 21
Figura 4 Características fundamentales de un agente 28
Figura 5 Arquitecturas de comunicación para LLM-MAS 29
Figura 6 Visión general del protocolo A2A 34
Figura 7 Tipos de paradigmas de comunicación 43

Figura 8 Implementación del algoritmo de chunking semántico basado en similitud del coseno 68
Figura 9 Arquitectura multiagente 78

Figura 10 Arquitectura del sistema multiagente implementado 97
Figura 11 Interfaz de usuario diseñada con gradio 101
Figura 12 Fragmento del reporte HTML con criticidad de cláusula 105
Figura 13 Comparación de la latencia por agente y equipo de hardware 109
Figura 14 Estadísticas descriptivas de los tiempos de procesamiento por modelo y dispositivo 111
Figura 15 Evaluación comparativa de modelos según métricas semánticas (G-Eval) 114
Figura 16 Rendimiento del modelo basado en la evaluación humana 118
Figura 17 Comparativa G-Eval vs. evaluación humana por modelo y métrica . . . 122

ÍNDICE DE TABLAS

Tabla I Tipos de sesgos en sistemas jurídicos basados en IA 22

Tabla II Sprint 01 - Infraestructura Base 70
Tabla III Sprint 02 - Agente Almacenador 71
Tabla IV Sprint 03 - Agente Analizador 72
Tabla V Sprint 04 - Agente Orquestador 73
Tabla VI Sprint 05 - Interfaz de Usuario 74
Tabla VII Sprint 06 - Integración y Estabilización 75
Tabla VIII Sprint 07 - Seguridad y Configuración 76
Tabla IX Sprint 08 - Testing y Calidad 77
Tabla X Sprint 09 - Documentación 78
Tabla XI Especificación de los agentes del sistema 79
Tabla XII Requisitos Funcionales del Sistema 85
Tabla XIII Requisitos No Funcionales del Sistema 86
Tabla XIV Métricas de precisión semántica para evaluación del análisis 91
Tabla XV Métricas de eficiencia operacional del sistema 93
Tabla XVI Características de los equipos de hardware utilizados en la evaluación . . 94
Tabla XVII Variables del análisis comparativo 95

Tabla XVIII Características del corpus de contratos SaaS 102
Tabla XIX Estadísticos descriptivos de latencia por agente, operación y equipo . . . 108
Tabla XX Medidas descriptivas del tiempo de análisis por modelo y entorno de ejecución 110
Tabla XXI Puntuaciones G-Eval por modelo: Fidelidad (F), Relevancia (AR), Precisión de Contexto (CP), Recall de Contexto (CR) 114
Tabla XXII Evaluación humana por modelo 118
Tabla XXIII Comparativa G-Eval vs. Evaluación humana por modelo y métrica 122
Tabla XXIV Puntuaciones G-Eval por modelo (valores originales) 140
Tabla XXV Evaluación humana por modelo (valores originales) 142
Tabla XXVI Comparativa G-Eval vs Evaluación humana (valores originales) 145

RESUMEN

Contexto. Los contratos SaaS plantean un reto para usuarios no especializados debido a su lenguaje jurídico técnico y complejas disposiciones en materia de propiedad de datos, seguridad y cumplimiento normativo. La creciente adopción de servicios en la nube ha agravado este problema, ya que los usuarios suelen aceptar condiciones contractuales extensas sin comprender plenamente derechos, obligaciones y prohibiciones.Los enfoques previos dependen de ajuste fino, limitando su flexibilidad y reproducibilidad entre distintas plataformas contractuales. **Objetivo.** El objetivo de esta investigación es desarrollar y evaluar una aplicación web multiagente coordinada mediante el protocolo Agent-to-Agent (A2A), que integre modelos de lenguaje para la extracción y análisis de cláusulas en contratos SaaS. **Método.** Se empleó un enfoque experimental cuantitativo con desarrollo iterativo en Scrum. La arquitectura incluye tres agentes especializados con Google ADK y CrewAI, que se comunican mediante A2A y utilizan los modelos Gemini 2.5 Flash Lite, Grok 4.1 Fast y Qwen3 VL 30B A3B Thinking vía OpenRouter. El sistema integra interfaz en Gradio y una base de datos vectorial Qdrant. **Resultados.** De los tres modelos empleados en el análisis, Grok 4.1 Fast fue el mejor, al ofrecer el mejor equilibrio entre calidad y eficiencia, con mayor fidelidad automatizada de 0,98, menor tiempo medio de respuesta de 70,89 s y menor desviación estándar de 9,70 s.Los tres modelos alcanzaron un Recall de Contexto perfecto de 1,00 en ambos entornos de hardware evaluados. **Conclusiones.** Los resultados demuestran la viabilidad de aplicar arquitecturas multiagente basadas en A2A al análisis automatizado de documentos jurídicos, proporcionando una solución funcional, escalable y reproducible para la comprensión de contratos SaaS en entornos tecnológicos y académicos.

Palabras clave: A2A, sistemas multiagente, Google ADK, CrewAI, términos de servicio, G-Eval

ABSTRACT

Context. SaaS contracts pose a significant challenge for non-specialized users due to their technical legal language and complex provisions regarding data ownership, security, and regulatory compliance. The growing adoption of cloud services has further worsened this problem, as users often accept extensive contractual terms without fully understanding their rights, obligations, and restrictions.Previous approaches rely on fine-tuning, which limits their flexibility and reproducibility across different contractual platforms. **Objective.** The goal of this research is to develop and evaluate a multi-agent web application coordinated through the Agent-to-Agent (A2A) protocol, integrating large language models for the extraction and analysis of clauses in SaaS contracts. **Method.** A quantitative experimental approach was used with iterative development under the Scrum framework. The architecture includes three specialized agents built with Google ADK and CrewAI, which communicate via A2A and leverage the Gemini 2.5 Flash Lite, Grok 4.1 Fast, and Qwen3 VL 30B A3B Thinking models through OpenRouter. The system integrates a Gradio-based interface and a Qdrant vector database. **Results.** Among the three models evaluated, Grok 4.1 Fast performed best, offering the optimal balance between quality and efficiency, with the highest automated faithfulness score of 0.98, the lowest average response time of 70.89 s, and the lowest standard deviation of 9.70 s. All three models achieved a perfect Context Recall of 1.00 in both hardware environments evaluated. **Conclusions.** The results demonstrate the feasibility of applying A2A-based multi-agent architectures to the automated analysis of legal documents, providing a functional, scalable, and reproducible solution for understanding SaaS contracts in both technological and academic settings.

Keywords: A2A, Multi-agent systems, Google ADK , CrewAI, Terms of Service, G-Eval

CAPÍTULO I

INTRODUCCIÓN

1.1 Planteamiento del problema

Los usuarios que contratan soluciones de software como servicio (SaaS) enfrentan desafíos significativos debido al desconocimiento de las condiciones establecidas en los contratos que regulan estas soluciones. Estos contratos, caracterizados por su lenguaje jurídico complejo y una extensa cantidad de disposiciones, incluyen términos críticos sobre la propiedad de datos, la seguridad, el cumplimiento normativo y los derechos de transición, que a menudo resultan difíciles de comprender o analizar eficientemente para los usuarios [1], [2]. Estas dificultades se sintetizan en la Figura 1.

No obstante, la cesión del control de los datos y la infraestructura de TI a proveedores de la nube aumenta la dependencia de los usuarios en las políticas y capacidades técnicas del proveedor, lo que puede generar preocupaciones sobre permisos, obligaciones y prohibiciones [1]. El principal desafío en la interpretación de documentos contractuales radica en distinguir entre relevancia y similitud, una complejidad que surge debido a los formatos estandarizados y las estructuras textuales repetitivas que se encuentran comúnmente en estos documentos [3].

Una solución para abordar los desafíos de analizar contratos SaaS es el desarrollo de una arquitectura multiagente basada en **Modelos de Lenguaje a Gran Escala** (LLM) de **código abierto**. Esta arquitectura utiliza el protocolo **Agent-to-Agent** (A2A) para sincronizar agentes que colaboran en la extracción, análisis y almacenamiento de las cláusulas relevantes. En este contexto, los sistemas cooperativos multiagentes está compuestos por varios agentes que, a través de sus interacciones buscar resolver tareas conjuntamente [4], estas interacciones coordinadas por el protocolo A2A, para identificar las distintas clausulas y almacenarlas en una base

datos para su posterior consulta.

Con esta arquitectura, los usuarios que contratan soluciones de software como servicio, podrán realizar consultas referentes a las cláusulas de los contratos SaaS que adquirieron, de esta forma mitiga los obstáculos significativos derivados de la complejidad y el volumen de los contratos, facilitando la debida actividad y el cumplimiento normativo [1].



Figura 1.
Análisis causal del problema SaaS

1.2 Objetivos

1.2.1 Objetivo General

Desarrollar una aplicación web para la extracción de cláusulas en contratos SaaS utilizando una arquitectura multiagente basada en el protocolo A2A y con el uso de LLMs que ofrezca soporte a las consultas contractuales y permita un análisis preciso para los usuarios.

1.2.2 Objetivos Especificos

- Analizar el protocolo A2A para comunicación entre agentes y las tecnologías relacionadas.

- Desarrollar una aplicación web para la extracción automática de cláusulas de contratos SaaS con LLMs utilizando una arquitectura multiagente con el protocolo A2A.
- Evaluar la aplicación web con métricas específicas para agentes de inteligencia artificial.

1.3 Alcance

El desarrollo de la aplicación web para la extracción de cláusulas en contratos SaaS se gestionará con SCRUM. El trabajo se organizará en sprints con entregas regulares, permitiendo una evolución controlada de la aplicación y una adaptación a los requisitos específicos. Antes del desarrollo, se investigará el marco teórico de las tecnologías relacionadas con el protocolo A2A, con el fin de establecer una base conceptual sólida que respalde el diseño y la implementación. Dicha aplicación se basará en una arquitectura multiagente utilizando el protocolo A2A y modelos LLMs, y será probada con 10 contratos SaaS reales de diferentes plataformas para garantizar su efectividad en contextos variados. El enfoque estructurado que ofrece esta metodología facilitará la ejecución ordenada de cada fase del desarrollo, desde el análisis inicial hasta la evaluación del sistema.

Después del desarrollo, se evaluarán las respuestas generadas por la aplicación utilizando métricas específicas para sistemas de inteligencia artificial, lo que permitirá medir su precisión, coherencia y capacidad de comprensión semántica.

Los resultados finales tienen como objetivo ofrecer una aplicación web que brinde soporte a las consultas contractuales mediante la extracción de cláusulas, permitiendo una revisión más precisa por parte de los usuarios, sobre aspectos como los permisos, prohibiciones y obligaciones [5].

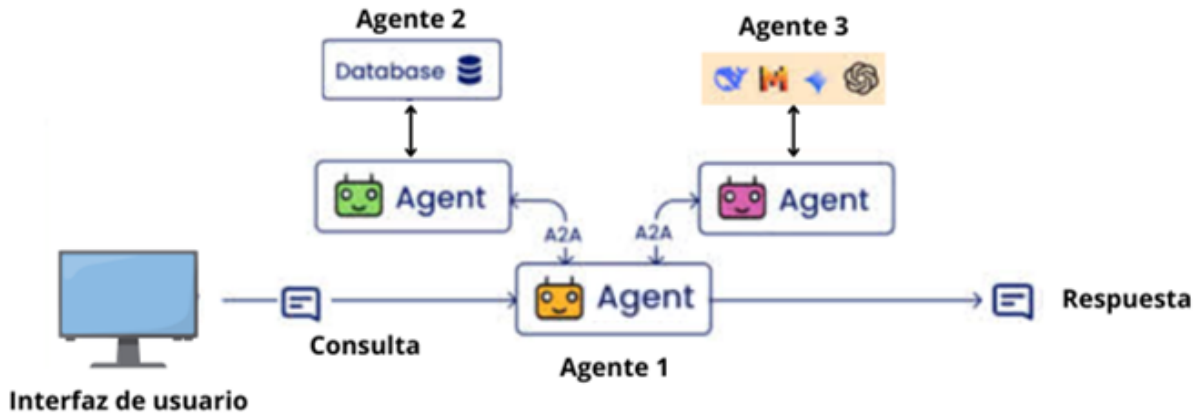


Figura 2.
Arquitectura general

La arquitectura en general se basa en una interacción entre el usuario y los diferentes agentes que tienen su determinada tarea a cumplir, coordinados por medio del protocolo A2A, como se muestra en la Figura 2, a continuación, se detalla cada uno de los componentes explicados en la figura:

- **Interfaz de Usuario:** El usuario realiza consultas a través de la interfaz amigable y en lenguaje natural, iniciando el flujo hacia los agentes.
- **Agentes IA:** Los agentes: El agente 1 coordina tareas al recibir la consulta del usuario, el agente 2 extrae cláusulas de contratos SaaS usando A2A, y el agente 3 (un LLM) analiza y responde. Estos agentes trabajan de forma autónoma, planifican y se adaptan en tiempo real a las instrucciones [5], [6].
- **A2A:** Mediante este protocolo, los agentes pueden interoperar entre sí para gestionar tareas, lo que facilita una comunicación fluida entre agentes de IA pertenecientes a distintos marcos y proveedores, y a su vez proporciona un mayor nivel de autonomía [7], [8]. Ya que está diseñado específicamente para apoyar la colaboración compleja entre agentes dentro del mismo ecosistema [9].
- **LLM:** El LLM procesa la consulta del usuario y las cláusulas recuperadas, generando una respuesta coherente y contextualizada basada en la información extraída de la base de datos [10].

- **Respuesta:** La aplicación web entrega una respuesta o información solicitada por el usuario en un formato natural y preciso basado en lo solicitado.

El diseño de la aplicación web asegura una interacción eficiente con el usuario, donde la colaboración de los diferentes agentes y el uso del LLM mejoran la respuesta a las consultas, asegurando que sean relevantes, permitiendo la automatización de la investigación y la asistencia inteligente [4], [6].

1.4 Justificación

El desarrollo de una aplicación web para la extracción de cláusulas en contratos SaaS mediante una arquitectura multiagente basada en el protocolo A2A y el uso de LLMs responde a una necesidad crítica en el entorno tecnológico moderno: facilitar la comprensión y el análisis de contratos complejos por parte de usuarios no especializados. Estos contratos, al estar redactados con un lenguaje jurídico técnico y contener disposiciones extensas, representan un desafío significativo para usuarios que adquieren soluciones SaaS sin un conocimiento legal profundo, lo que pone en riesgo la seguridad de sus datos, el cumplimiento normativo y su capacidad de tomar decisiones informadas [1], [5].

Esta propuesta busca aporta una solución innovadora y técnica al automatizar la extracción de cláusulas contractuales relevantes, utilizando IA distribuida mediante agentes cooperativos [4]. La implementación de una arquitectura multiagente, coordinada a través del protocolo A2A, permite dividir las tareas de extracción, análisis y clasificación en componentes especializados, aumentando la eficiencia, escalabilidad y precisión del proceso [6], [11]. La formalización de la comunicación que provee A2A, junto con su soporte nativo para la gestión de memoria y la coordinación de objetivos, constituye la base para orquestar un flujo de trabajo multiagente confiable, donde el contexto del contrato se mantiene a través de las distintas etapas de procesamiento desde la extracción de texto hasta la generación de respuestas precisas a las consultas de los usuarios [12].

Este desarrollo está tecnológicamente justificado al implementar la integración de LLMs con arquitecturas multiagente, utilizando el protocolo A2A para asegurar interoperabilidad. Su aplicabilidad queda demostrada al extrapolar el éxito de agentes en dominios de alta precisión como salud y finanzas al análisis contractual, un campo de similar complejidad y exigencia [13].

1.4.1 Beneficios

Desde el punto de vista académico, el proyecto representa un avance en la aplicación de técnicas de inteligencia artificial distribuida, combinando agentes inteligentes y modelos LLMs de código abierto, lo cual enriquece la literatura sobre sistemas inteligentes aplicados al derecho digital y la gestión de contratos electrónicos [11].

En el ámbito investigativo, la propuesta explora un enfoque novedoso para tratar contratos con aspectos técnicos y legales como los SaaS, introduciendo una arquitectura que puede ser replicada o extendida a otros dominios contractuales [5], [14]. Además, permite evaluar el rendimiento de diferentes modelos LLMs en tareas específicas de procesamiento jurídico.

A nivel social y educativo, los usuarios finales podrán acceder a herramientas que les permitan comprender fácilmente sus derechos, deberes y restricciones al contratar un servicio en la nube [3]. Fortaleciendo la transparencia y reduce la brecha de conocimiento jurídico-técnico, promoviendo el empoderamiento digital al servir de recurso para estudiantes en IA aplicada al comercio [15].

En el plano económico y político, el proyecto facilita el cumplimiento normativo y contractual en entornos regulados, lo cual puede reducir sanciones legales por incumplimientos, fomentar la adopción responsable de servicios SaaS y mejorar la gobernanza de datos en las organizaciones [5], [13].

En conclusión, esta investigación no solo resuelve un problema concreto que enfrentan los usuarios de servicios SaaS, sino que también genera beneficios tangibles y sostenibles en múltiples dimensiones, desde lo técnico hasta lo social, consolidándose como una propuesta con alto

impacto práctico y académico en el campo de la ingeniería de software y la inteligencia artificial aplicada [3].

CAPÍTULO II

MARCO TEÓRICO

Este capítulo recoge los fundamentos conceptuales y antecedentes que sostienen la investigación, con el propósito de situar al lector en los principios y enfoques necesarios para comprender el estudio.

2.1 Antecedentes

Los contratos constituyen instrumentos legales fundamentales para regular relaciones personales, comerciales y gubernamentales. No obstante, su revisión sigue siendo un proceso complejo que demanda tiempo, experiencia jurídica y la capacidad de interpretar terminología especializada, cláusulas ambiguas o incluso contradictorias [3]. Estas dificultades evidencian la necesidad de herramientas que faciliten su comprensión y reduzcan la dependencia de la revisión manual, especialmente para personas sin formación legal.

En los últimos años, el análisis automatizado de contratos ha dado pasos importantes. La llegada de los grandes modelos de lenguaje, sumada al desarrollo de arquitecturas multiagente y protocolos de comunicación especializados, ha cambiado la manera en que se extrae y organiza la información contractual en entornos profesionales [3]. En los contratos SaaS, en particular, la coordinación de agentes inteligentes ha pasado de ser una opción a convertirse en una necesidad. Todo esto ha abierto una línea activa de investigación orientada a mejorar la precisión y eficiencia del análisis jurídico, con implicaciones reales para la gestión de documentos legales.

Por ejemplo, uno de los desarrollos más representativos es el propuesto por Seabra et al. (2024), el cual introduce el sistema *Contrato360*, un enfoque híbrido que combina agentes inteligentes, LLMs y técnicas de recuperación multimodal. Este sistema no solo integra información

proveniente de PDF y bases de datos, sino que además emplea RAG y Text-to-SQL para ofrecer respuestas más precisas. La administración del flujo de trabajo por parte de agentes autónomos permite seleccionar la técnica adecuada según el tipo de pregunta, lo que resulta particularmente útil en consultas complejas o indirectas [3].

La evaluación con especialistas mostró un nivel de acierto bastante alto, aunque, en consultas que exigían un análisis semántico más profundo, aparecieron algunas respuestas algo incompletas. Aun así, el sistema logra integrar fuentes diversas con bastante solidez, lo que, en conjunto, supone un paso adelante relevante en la gestión contractual moderna [3].

En esta misma línea, Raptopoulos et al. (2025) proponen *PAKTON*, un marco multiagente de código abierto que profundiza en la idea de agentes especializados. Su arquitectura tripartita (Archivista, Investigador e Interrogador) introduce una forma estructurada y colaborativa de analizar contratos legales extensos. Lo verdaderamente llamativo es que, además de recuperar información, busca justificar sus respuestas, generando informes explicativos que facilitan el escrutinio jurídico [5].

Los resultados muestran que PAKTON supera de forma sostenida a modelos generales y especializados como GPT-4o y SaulLM tanto en precisión (F1) y recall en tareas de inferencia (ContractNLI) como en recuperación (LegalBench-RAG). Los expertos humanos también lo prefieren en criterios como explicabilidad e integridad de los informes. Que ambas evaluaciones técnica y humana apunten en la misma dirección refuerza el valor de la colaboración entre agentes para producir resultados más transparentes y confiables [5].

De manera complementaria, Liao et al. (2025) presentan *AgentMaster*, un framework modular de sistemas multiagente que integra los protocolos A2A y MCP. Este enfoque permite descomponer consultas complejas en subtareas que son ejecutadas de forma coordinada por agentes especializados en dominios como SQL, recuperación de información (IR) y análisis de imágenes [6].

Los resultados del estudio demuestran un alto rendimiento en tareas estructuradas, con mé-

tricas de precisión y consistencia notables en consultas SQL y recuperación de información; sin embargo, el sistema muestra variabilidad en tareas abiertas y de generación lingüística libre, como las realizadas por el agente general y el agente de imágenes [6].

A pesar de ello, la coordinación efectiva entre agentes y la capacidad para sintetizar respuestas a partir de múltiples fuentes modales evidencian que la integración de protocolos estandarizados es un avance significativo hacia arquitecturas más robustas y escalables, con implicaciones prometedoras para dominios que requieren análisis multimodal y procesamiento colaborativo, como el contractual [6].

También resulta pertinente destacar el trabajo de Zin et al. (2023), quienes en el proyecto titulado *Information Extraction from Lengthy Legal Contracts: Leveraging Query-Based Summarization and GPT-3.5*, plantean un enfoque no supervisado para la extracción de información en contratos extensos. Su propuesta, basada en un proceso de dos pasos (resumen orientado por consultas y posterior extracción mediante GPT-3.5), responde de manera ingeniosa a las limitaciones de tokens de los modelos y a la escasez de datos anotados. Los resultados mostraron que este método superó a modelos supervisados como RoBERTa y DeBERTa, con un F1 de 0.76 frente a 0.73, y alcanzó una precisión del 88 % en la recuperación de oraciones relevantes mediante BM25. El enfoque funcionó bien tanto para identificar entidades breves como cláusulas extensas, incluso en contratos que superaban los límites de procesamiento de GPT-3.5 [14]. Esto sugiere que las estrategias híbridas pueden ofrecer soluciones eficientes sin necesidad de grandes volúmenes de datos etiquetados, una limitación recurrente en el ámbito jurídico.

Por su parte, el trabajo desarrollado por Kim et al. (2025) en el proyecto titulado *Development of an Automated Construction Contract Review Framework Using Large Language Model and Domain Knowledge*, se centra en la automatización de la revisión de contratos de construcción, introduciendo un sistema que combina LLMs con conocimiento experto estructurado. El uso de un marco basado en Formularios de Contrato Estándar (SCF), junto con un mecanismo de recuperación híbrido en dos etapas y generación de respuestas basada en referencias, permite obtener análisis altamente precisos y confiables. Los resultados demostraron que este sistema superó significativamente a los enfoques de LLM base, LLM aumentado por búsqueda y RAG

básico, logrando una mejora del 52.6 % en la recuperación de contexto y del 48.3 % en la fidelidad, además de alcanzar la mayor puntuación promedio (0.828) con el desempeño más estable. Aunque persisten desafíos en razonamiento complejo, los resultados superan claramente a las aproximaciones puramente generativas, mostrando que la incorporación de conocimiento especializado es fundamental cuando se trabaja con documentos legales con alto nivel de tecnicismo [15]

En conjunto, estos estudios evidencian una evolución hacia sistemas híbridos y multiagente que no solo recuperan información contractual, sino que también la organizan y validan, sentando las bases para soluciones más eficientes y confiables en el análisis automatizado de contratos SaaS.

2.2 Contratos SaaS y Análisis Jurídico

Un contrato SaaS no es un único documento, es un conjunto de lineamientos contractuales que están interconectados, los cuales especifican los términos vinculantes tanto para el proveedor del servicio como para el cliente. Este tipo de acuerdo es dinámico y posee un alcance transnacional [16]. En él se detallan los derechos, obligaciones y prohibiciones que ambas partes deben acata, aunque dichos términos pueden modificarse en cualquier momento, sin importar el lugar donde se encuentren las partes.

Por otro lado, un análisis jurídico resulta de vital importancia para recopilar la totalidad de los documentos contractuales, con el fin de identificar todos los términos que regulan el uso de una solución SaaS especifica [16]. Este análisis puede evaluar si se está cumpliendo con las disposiciones contractuales de forma legal y adecuada, evitando posibles sanciones o pagos económicos por incumplimientos.

2.2.1 Estructura y componentes críticos de contratos SaaS

En general, los contratos SaaS presentan una estructura común orientada a regular el servicio, el uso del software y la relación entre proveedor y cliente. Inician con una introducción que identifica a las partes, establece la vigencia e incluye definiciones clave, seguida de la descripción del servicio, donde se especifican su alcance, funcionalidades, limitaciones y las condiciones de la licencia y derechos de uso [16], [17].

Asimismo, incorporan cláusulas sobre duración, renovación y terminación, junto con los términos de precios, facturación y pagos, y los Acuerdos de Nivel de Servicio (SLA) que fijan las garantías de disponibilidad y soporte. Finalmente, regulan la propiedad y protección de los datos, la seguridad, el cumplimiento normativo, el soporte y mantenimiento, la confidencialidad, las indemnizaciones, las limitaciones de responsabilidad y las disposiciones generales referentes a modificaciones, cesión, fuerza mayor y jurisdicción aplicable [16], [17].

Todos los elementos que conforman un contrato SaaS son importantes, pero existen secciones que el usuario que adquiere este tipo de servicios debería tener muy presentes, ya que son clausulas cruciales en caso de disputas, reclamos o posibles problemas legales. Estos son:

- **SLA:** Define calidad, disponibilidad y remedios [17].
- **Propiedad de datos, seguridad y privacidad:** Protege datos del cliente y su cumplimiento legal [16], [17].
- **Confidencialidad e indemnizaciones:** Regula las responsabilidades del incumplimiento de la condición de confidenciatos legales o vulneraciones de derechos. [17].
- **Limitaciones de responsabilidad:** Establecer límites a la exposición económica por ambos lados partes frente a posibles daños [17].
- **Licencia y derechos de uso:** Garantiza el uso legítimo del software y regula restricciones [16], [17].

2.2.2 Tipología de cláusulas legales

Los contratos SaaS incluyen diversos tipos de cláusulas legales que regulan la relación que existe entre el proveedor y el usuario del servicio. Estas cláusulas suelen abordar aspectos relacionados con los derechos (permisos), obligaciones, prohibiciones y limitaciones de responsabilidad de ambas partes, garantizando así un marco contractual equilibrado y seguro.

- **Derechos (Permisos)**

  En un contrato SaaS, el proveedor suele contar con la facultad de modificar unilateralmente los términos del servicio, así como las políticas de seguridad y de uso aceptable, además de reservarse el derecho de suspender o terminar las cuentas de usuarios finales o incluso el contrato principal en caso de incumplimiento de dichas políticas citeLynn2021.

  Por su parte, el usuario mantiene el derecho a conocer dónde se almacenan y cómo se utilizan sus datos, así como a ser notificado oportunamente sobre cualquier cambio en el acuerdo o en las condiciones del servicio [17].

- **Obligaciones**

  En los contratos SaaS, el SLA establece obligaciones específicas para el proveedor, quien debe garantizar un nivel de servicio definido mediante métricas de calidad y, en caso de incumplimiento, compensar al cliente por el rendimiento inferior al acordado [17].Asimismo, bajo el GDPR (Reglamento General de Protección de Datos), el proveedor está obligado a actuar únicamente siguiendo las instrucciones del cliente, considerado el controlador de los datos, y debe asegurar la confidencialidad, integridad y disponibilidad de la información personal. Finalmente, también se le exige disponer de procesos claros para la preservación, eliminación o transferencia de los datos del cliente una vez concluido el servicio

- **Prohibiciones**

  El contrato también incorpora Políticas de Uso Aceptable (AUP), las cuales establecen prohibiciones tanto para el proveedor como para el usuario. En el caso del proveedor, se

prohíbe la exposición de información sensible almacenada en la plataforma, especialmente con fines fraudulentos, así como su venta a terceros. Por parte del usuario, se restringen actividades ilegales como el hackeo, la piratería o el alojamiento de contenido ofensivo; además, se limita el uso del servicio en contextos de alto riesgo como el transporte aéreo o instalaciones nucleares donde una falla podría generar consecuencias graves. Asimismo, se prohíbe cualquier comportamiento abusivo hacia el personal del proveedor [17].

- **Limitaciones de responsabilidad**

  Una de las cláusulas más críticas para el proveedor es la limitación de responsabilidad. Los proveedores intentan minimizar o excluir su responsabilidad por cualquier perdida (directa o indirecta) derivada de una interrupción del servicio, una brecha de seguridad o la pérdida de datos [17].

  En lugar de pagos económicos por daños, la solución típica se limita a créditos de servicio, cuyo uso a menudo está restringido. Esta cláusula busca limitar el riesgo financiero del proveedor ante un incidente, trasladando gran parte del riesgo al cliente [17].

2.3 Modelos de lenguaje a gran escala

Los LLMs se han vuelto una herramienta muy útil para analizar contratos legales, incluidos los de tipo SaaS. Según Zin et al. [14], estos LLMs nos ofrecen tres ventajas importantes a destacar:

- Pueden entender el lenguaje jurídico sin necesidad de entrenamiento específico, ya que aprovechan el conocimiento adquirido durante su preentrenamiento [14].
- Procesan documentos extensos mediante un enfoque estratégico que identifica y analiza únicamente las secciones relevantes [14].
- Nos permiten extraer al mismo tiempo información puntual, como lo son las fechas o nombres, y cláusulas más complejas, como los términos de servicio o SLA [14].

Estas capacidades se sustentan en la naturaleza avanzada de los LLMs, que son modelos preentrenados con miles de millones de parámetros. Además de comprender y generar texto, también exhiben habilidades emergentes, como el aprendizaje en contexto, el seguimiento de instrucciones complejas y el razonamiento en múltiples pasos [18], lo que amplía considerablemente su utilidad en tareas especializadas.

En esencia, los LLMs transforman una tarea que tradicionalmente requería abogados especializados y meses de trabajo en un proceso eficiente basado en consultas dirigidas a una inteligencia artificial que ya comprende la estructura y el funcionamiento de los contratos [14].

No obstante, estos modelos presentan limitaciones inherentes, como la posible generación de información errónea, su naturaleza estática y los altos costos computacionales asociados tanto a su entrenamiento como a su despliegue [18]. En este contexto, y con el propósito de mitigar estas dificultades, han surgido técnicas complementarias como RAG (Generación Aumentada por Recuperación), que permite enriquecer las respuestas mediante la recuperación de información externa. Asimismo, más recientemente, han cobrado relevancia los agentes de IA, los cuales permiten a los modelos acceder de manera dinámica a herramientas y fuentes externas de conocimiento actualizado, ampliando así sus capacidades de razonamiento y ejecución de tareas.

2.3.1 Arquitecturas de LLMs open-source

El panorama de las arquitecturas modelos ha cambiado notablemente en los últimos años, tanto en su popularidad como en sus capacidades. Modelos como GPT-4 y Claude lideran esta evolución por su potencia y versatilidad. Paralelamente, la aparición de modelos de código abierto demuestra que estas alternativas pueden ser altamente competitivas e incluso superar a algunos modelos cerrados [19]. Entre estos destacan los siguientes:

- **Llama 2**

  Los modelos Llama 2 fueron publicados de forma abierta para uso comercial y de in-

vestigación, acompañados de lineamientos de uso seguro y restricciones contra aplicaciones malintencionadas. Están construidos sobre una arquitectura Transformer con prenormalización y embeddings optimizados, y utilizan un vocabulario de 32.000 tokens [20].

Una de sus mejoras más destacadas es la ampliación de la ventana de contexto de 2.000 a 4.000 tokens, lo que facilita el manejo de conversaciones y documentos más extensos. A su vez, la familia incluye variantes de 7B, 13B, 34B y 70B parámetros, todas orientadas a procesar contextos más largos con una mayor eficiencia [20].

Gracias a estas características, se convierten en una opción bastante sólida tanto para sistemas conversacionales como para despliegues en entornos productivos que requieren un modelo abierto y de alto rendimiento.

- **Mistral**

  La familia Mistral ha cobrado protagonismo en el ecosistema de la IA por su apuesta por modelos ligeros, eficientes y de alto rendimiento. Dentro de sus variantes (Mistral 8x7B, 8x22B, Large 2 y Small) Mistral 7B es el modelo base de la serie y uno de los más utilizados. Parte de la arquitectura Transformer, pero incorpora mejoras orientadas a optimizar la eficiencia y acelerar la inferencia, lo que le permite manejar secuencias largas, reducir el consumo de memoria y responder con mayor rapidez [21].

  Con esas capacidades, Mistral 7B se presenta como una alternativa competitiva dentro del ecosistema open-source, capaz de superar incluso a modelos de mayor tamaño en diversas tareas [21]. Un resultado que muestra lo que puede lograrse con una ingeniería bien pensada, sin necesidad de escalar el tamaño del modelo.

- **Falcon**

  La familia Falcon emplea una arquitectura decoder-only basada en Transformer, optimizada para la escalabilidad en rendimiento, datos y hardware. Su principal aporte es la extensión del mecanismo de atención multiquery hacia atención multigrupo, lo que reduce de forma notable el consumo de memoria durante la generación y mejora la eficiencia en entornos de alta demanda [22].

Los modelos de la serie Falcon (Falcon-7b, 40b y 180b) aumentan gradualmente en numero parámetros, pero todas estas unidades comparten la misma ventana de contexto fija: 2.048 tokens que los limita frente a secuencias largas. Aun con esa restricción, Falcon mantiene un una buena prestación competitiva para tareas a gran escala aunque el precio de la inferencia puede ser grande. convertirse en un obstáculo dependiendo del caso de uso [22].

- **DeepSeek**

  Este modelo ha emergido como un actor importante en el ámbito de la IA sobre todo en lo que respecta al entrenamiento de modelos con eficiencia de costos, gracias a esto han liberado modelos que son capaces de igualar o supera el rendimiento de modelos cerrados, Entre sus propuestas más destacadas se encuentran DeepSeek-V2, DeepSeek-V3 y DeepSeek-R1, estos centrados en mejorar el rendimiento sin incrementar de forma desproporcionada los recursos necesarios para entrenar e inferir [23].

  Estos modelos han demostrado un desempeño sobresaliente en tareas de razonamiento y programación, apoyados en un proceso de entrenamiento transparente y cuidadosamente curado. Este enfoque ha permitido a DeepSeek construir modelos competitivos, eficientes y con una base de entrenamiento verificable, consolidándose como una alternativa sólida dentro del ecosistema de modelos de código abierto [23].

- **Qwen**

  Los modelos Qwen se basan en una arquitectura optimizada para procesar secuencias largas de forma eficiente, apoyándose en estrategias de entrenamiento progresivo que mejoran su rendimiento y reducen tiempos de cómputo. Estas optimizaciones también aceleran la inferencia, lo que los hace adecuados para aplicaciones en tiempo real y escalables en contextos tanto académicos como industriales [23].

  Dentro de la familia sobresalen dos modelos. Qwen2.5-Max destaca por su rendimiento equilibrado en razonamiento, conocimiento general y programación, mientras que Qwen3-235B-A22B compite directamente con modelos de última generación como DeepSeek y Gemini. En términos generales, la serie Qwen combina versatilidad y solidez en un rango

amplio de dominios [23].

2.3.2 Fine-tuning para dominio jurídico

El fine-tuning en el dominio jurídico implica tomar un modelo de lenguaje ya entrenado y ajustarlo a tareas legales específicas usando datos etiquetados del área. Su propósito es reforzar la comprensión del modelo sobre la terminología y los conceptos propios del derecho, los que permite un desempeño más preciso en tareas especializadas [24].

Esta adaptación se logra entrenando el modelo con textos jurídicos como sentencias, contratos, doctrina y consultas legales, lo que le permite avanzar de un conocimiento general a una comprensión más profunda y contextualizada del lenguaje legal [25].

Gracias a ello, estas tecnologías pueden identificar cláusulas abusivas, apoyar procesos de verificación y contribuir al cumplimiento normativo, reforzando la transparencia hacia los usuarios y la utilidad del fine-tuning para mejorar la precisión en la revisión automatizada de documentos legales [24].

- **Técnicas de adaptación**

  - **Supervised Fine-Tuning (SFT):** Técnica que adapta un modelo preentrenado a tareas específicas mediante aprendizaje supervisado. Aprovecha el conocimiento adquirido durante el preentrenamiento y ajusta el modelo con un conjunto reducido de datos etiquetados, ya sea para una sola tarea o bajo un enfoque multitarea [25].
  - **Parameter Efficient Fine-Tuning (PEFT):** Técnica que ajusta solo un subconjunto del modelo, reduciendo el consumo de recursos computacionales y de memoria durante el entrenamiento. Esto permite aplicar ajuste fino en entornos con recursos limitados, manteniendo un buen desempeño incluso en tareas complejas [25].
  - **Reinforcement Learning from Human Feedback (RLHF):** Método que requiere

intervención humana en el entrenamiento para ajustar el comportamiento del modelo, logrando respuestas precisas, seguras y alineadas con la intención del usuario. Esto incrementa el control sobre el modelo y reduce resultados no deseados [25].

- **Reinforcement Fine-Tuning (RFT):** Técnica que incorpora aprendizaje por refuerzo durante el ajuste fino para especializar al modelo en tareas específicas y optimizar su desempeño. Bajo este enfoque, el modelo aprende a generar respuestas de mayor calidad mediante recompensas o penalizaciones según la corrección de sus salidas [25].
- **Pre-Training Fine-Tuning (PT-FT):** Proceso que continúa el entrenamiento de un modelo previamente preentrenado utilizando datos específicos del dominio, con el fin de reforzar su desempeño en tareas especializadas y reducir errores derivados de conocimientos generales [25].
- **Full Fine-Tuning (FFT):** Método que modifica todos los parámetros del modelo para maximizar su rendimiento en una tarea específica. Aunque permite una adaptación completa, requiere alta capacidad computacional y puede generar sobreajuste cuando los datos etiquetados son limitados [25].
- **LoRa y Adapter:** Estrategias que ajustan solo una pequeña porción de los parámetros del modelo, reduciendo significativamente el costo computacional y mejorando el entrenamiento sin sacrificar el rendimiento ni la calidad de los resultados [24], [25].
- **Instruction Tuning:** Proceso en el que el modelo se entrena con un conjunto variado de instrucciones vinculadas a distintas tareas, mejorando su capacidad para comprender y ejecutar tareas complejas y aumentando su adaptación a nuevos contextos [25].

- **Prompt engineering aplicado al derecho**

  La ingeniería de prompts se define como un marco metodológico orientado a la construcción y diseño de instrucciones que busquen optimizar el desempeño de un modelo de lenguaje grande en una tarea en específico [26]. Esta técnica ofrece una mayor flexibilidad

para adaptarse a diferentes tareas y sobre todo no requiere de un entrenamiento adicional.

Dentro del ámbito jurídico, esta disciplina se adapta como ingeniería de prompts legales, cuyo objetivo es diseñar, evaluar y refinar instrucciones efectivas para aplicaciones de procesamiento de lenguaje natural en derecho [27]. Esto permite utilizar los modelos mediante consultas en lenguaje natural para realizar tareas como anotación de datos, recuperación de información o respuestas a preguntas jurídicas.

- **Estructura de un prompt en el contexto legal**

  En el ámbito jurídico y en la interacción con sistemas de IA, la claridad, la lógica y la estructura son fundamentales. Los marcos de razonamiento lógico, ampliamente utilizados en la práctica legal y académica, no solo funcionan como herramientas pedagógicas, sino también como modelos para generar respuestas precisas y coherentes en entornos asistidos por IA. Entre ellos destaca el método IRAC (Issue, Rule, Application, Conclusion), uno de los más consolidados para organizar el razonamiento legal y que resulta especialmente útil para construir prompts destinados a modelos de lenguaje, pues guía al sistema mediante un proceso de análisis claramente definido [28].

  El enfoque IRAC es uno de los marcos más utilizados en el razonamiento jurídico tradicional y también resulta útil para la formular prompts efectivos en modelos de lenguaje. Un ejemplo de ello es el que plantea Fangyi Yu et al. [28], donde le solicita al modelo de IA que **piense como un abogado** mediante la combinación de componentes como: instrucción + enfoque + premisa + hipótesis + "¿Verdadero o Falso?".

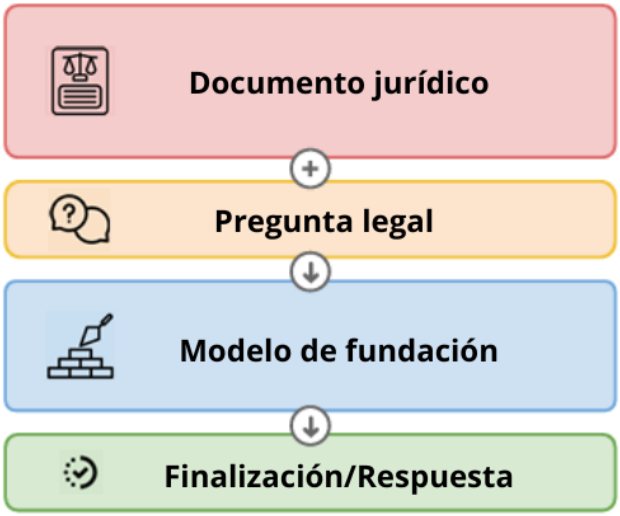


Figura 3.
Pila de prompting legal (adaptado de [27])

La Figura 3 muestra la arquitectura típica de un sistema de prompting legal: el modelo base integra el contexto documental y la pregunta jurídica para generar una respuesta estructurada, tal como se logra al guiarlo con el marco IRAC [27]. Esto como lo propone Fangyi Yu et al.

2.3.3 Limitaciones en extracción de cláusulas

La extracción de información en documentos legales busca obtener conocimiento estructurados como entidades, relaciones y eventos a partir de textos no estructurados. En el ámbito jurídico, la extracción de cláusulas ha adquirido relevancia dentro del NLP (Procesamiento de Lenguaje Natural) debido a la creciente demanda de obtener información jurídica de forma eficiente por parte de usuarios, profesionales de derecho y organizaciones [29].

No obstante, los documentos legales presentan características únicas que dificultan su procesamiento, lo que evidencia la necesidad de aplicar técnicas avanzadas de NLP para superar los desafíos del área [29]. En este contexto, la extracción de cláusulas legales enfrenta limitaciones como sesgos, contexto limitado e interpretación ambigua, factores que complican la obtención de resultados precisos en los sistemas automáticos.

- **Sesgo y Tipos de sesgos**

  Según Damith P et al. [29], un sesgo refleja una distribución desigual en los recursos de investigación que favorece desproporcionadamente a ciertos elementos mientras otros quedan subrepresentados. Esa asimetría restringe la generalización, la equidad y el avance en la extracción de información legal, abriendo una brecha entre los sistemas desarrollados y la diversidad del mundo real. En la Tabla I se presentan los principales tipos de sesgos identificados en este contexto.

Tabla I.
TIPOS DE SESGOS EN SISTEMAS JURÍDICOS BASADOS EN IA

Tipo de sesgo	Descripción
Sesgo de idioma	Surge debido a la falta de conjuntos de datos lingüísticos, lo que frena el desarrollo de tecnologías jurídicas basadas en procesamiento del lenguaje natural. Esto reduce la capacidad de los sistemas para operar eficazmente en contextos legales con distintos idiomas y culturas [30].
Sesgo ético	Constituye una preocupación central en el uso de sistemas de IA, ya que la sobreconfianza en estos modelos puede afectar la equidad, la transparencia y favorecer la perpetuación de prejuicios [30]. Este sesgo también aparece cuando se sobreestiman sus capacidades y se omite la revisión necesaria, olvidando que la IA es una herramienta de apoyo y no sustituto del juicio humano.

Tipo de sesgo	Descripción
Sesgo de datos	Hace referencia al sesgo ligado a la fuente y composición de los datos de entrenamiento. Surge porque gran parte de la investigación se basa en corpus legales propietarios o personalizados que no representan la diversidad del panorama legal global, dejando de lado otros conjuntos de datos y estudios igualmente relevantes [29], [30].
Sesgo de ubicación/jurisdicción	Se evidencia en los conjuntos de datos usados para el NLP, donde ciertos sistemas legales están sobrerrepresentados. Esto limita la aplicabilidad y capacidad de generalización de los modelos hacia otras jurisdicciones y regiones, pasando por alto la diversidad jurídica existente a nivel global [29], [30].

- **Contexto Limitado**

  El contexto limitado constituye el principal obstáculo en la extracción de cláusulas, ya que la mayoría de los documentos legales presentan una extensión considerable y su información se encuentra dispersa a lo largo del texto, lo que dificulta que los métodos especializados identifiquen y extraigan con precisión la información relevante [29].

  Los modelos preentrenados suelen presentar dificultades en el ámbito legal debido a su capacidad limitada para procesar contextos extensos; en consecuencia, no logran visualizar el documento completo de manera simultánea, lo que impide comprender adecuadamente las relaciones distantes o el contexto globa [29].

  Según Weiwen Xu et al.[31], la mayoría de los modelos de lenguaje tienden a romper las correlaciones entre cláusulas, ya que dividen los documentos en fragmentos más pequeños por separado. Sin embargo, este proceso de segmentación tiene un impacto negativo ya que dificulta establecer las conexiones entre cláusulas ubicadas en diferentes secciones del texto legal.

Como solución práctica, algunos investigadores han utilizado la técnica de la ventana deslizante, la cual divide el texto en fragmentos para permitir que el modelo pueda procesarlo, aunque esto puede provocar la pérdida de la visión general del contexto global del documento [29].

- **Interpretación Ambigüa**

  Los documentos legales, en la práctica, suelen presentar bastantes ambigüedades y una fuerte dependencia del contexto, lo que los distingue claramente del lenguaje común. A la par, su complejidad lingüística y el uso de términos algo vagos hacen que el análisis con modelos de NLP no sea tan directo como parece, ya que el significado puede cambiar según el caso o incluso la jurisdicción [30].

  Todo esto, en conjunto, termina representando un obstáculo importante para la extracción automática de información, especialmente cuando se trata de identificar y entender cláusulas específicas.

  Para Weiwen Xu et al. [31], existen dos fuentes principales de la ambigüedad en los textos legales. La primera está relacionada con los poco claros términos legales y la falta de precisión. Puede crear confusión; en consecuencia, proponen que se establezca una Relación Término Definición (TDR). Que ponga en correspondencia a cada uno de los términos y su significado.

  Por otra parte, la segunda fuente de ambigüedad entre las cláusulas proviene de aquellos tipos que presentan una estructura o formato muy similar, como ocurre con “Fecha de Vencimiento” y “Fecha del Acuerdo”. Esta similitud, a su vez, dificulta su diferenciación automática, ya que ambos tipos comparten características textuales que pueden inducir a confusión durante el proceso de clasificación.

2.4 Desafíos en la interpretación automatizada de lenguaje jurídico

El NLP, o procesamiento del lenguaje natural, constituye un área dentro de la inteligencia artificial dedicada a permitido a los sistemas comprender, analizar y generar lenguaje humano

en forma de texto o voz, facilitando la interacción con las personas mediante lenguaje natural [32].

No obstante, su aplicación en ámbitos especializados como el lenguaje jurídico presenta importantes desafíos técnicos y conceptuales, ya que la interpretación automatizada de este tipo de lenguaje exige modelos capaces de enfrentar múltiples dificultades propias de su complejidad, las cuales se agrupan en diversas áreas clave [32].

- **Ambigüedad del lenguaje legal:** La naturaleza vaga y abierta del lenguaje legal, obliga a los modelos a resolver ambigüedades contextuales presentes en los contratos [33]. Interpretar una cláusula requiere entender el texto, la intención de las partes y su contexto.
- **Dificultad en la recuperación de información:** La recuperación de la información relevante es un proceso complejo, ya que requiere identificar entre los distintos casos que solo mencionan un concepto y aquellos que lo analizan en profundidad [33]. Esta tarea continúa siendo limitada incluso para los modelos más avanzados.
- **Sesgos en datos y bases jurídicas:** Los sesgos presentes tanto en los datos de entrenamiento como en las bases jurídicas pueden influir directamente en la forma en que el modelo interpreta la información [33]. Además, la fragilidad de estos sistemas facilita que un error o sesgo se replique y genere fallos sistemáticos [34].
- **Limitaciones del contexto disponible:** La limitada naturaleza del contexto: la cantidad de datos contextuales disponibles para los Los modelos que suelen manejar son insuficientes con el volumen de casos y los matices propias del ámbito jurídico [33]. Esta es una vía para que ellos produzcan información sin fundamento con consecuencias de errónea interpretación con efectos legales/económicos.
- **Evolución normativa y moral:** La constante actualización del marco legal y también del ámbito moral representa un desafío bastante importante. Y es que un sistema entrenado con datos legales del pasado no se adapta tan fácilmente a nuevas interpretaciones o a criterios jurídicos que van surgiendo, lo que, al final, puede dar lugar a análisis desactualizados o que no terminan de captar los matices de una norma vigente [34].

2.5 Sistemas Multiagente (MAS)

Los LLMs representan un avance fundamental en la inteligencia artificial, demostrando capacidades notables en comprensión y generación de lenguaje. Sin embargo, su arquitectura monolítica y esencialmente autorregresiva impone límites para aplicaciones complejas del mundo real. Estos incluyen la tendencia a generar alucinaciones, un razonamiento predominantemente lineal que dificulta la iteración profunda, y restricciones prácticas de eficiencia y escalado [35].

Para superar estas barreras estructurales y desbloquear el verdadero potencial de los LLMs en tareas que exigen precisión, coordinación y análisis en profundidad, se recurre al paradigma de los MAS. Los MAS son arquitecturas compuestas por múltiples agentes inteligentes que cooperan para resolver problemas complejos y distribuidos que superan las capacidades de un solo agente. En estos sistemas, cada agente posee roles, habilidades y autonomía para percibir, razonar y actuar dentro de un entorno compartido, interactuando mediante mecanismos estandarizados de comunicación, coordinación y negociación [10], [35].

La aplicación de este paradigma da lugar a los Sistemas Multiagente basados en LLM (LLM-MAS), que no reemplazan al modelo individual, sino que lo potencian mediante inteligencia colectiva. Inspirados en la colaboración humana, estos sistemas distribuyen tareas entre múltiples agentes especializados (cada uno potenciado por un LLM) y coordinados por un mecanismo de orquestación [35].

Esta transformación de un modelo monolítico en un ecosistema interactivo ofrece ventajas como la mitigación de errores mediante verificación cruzada y debate, la distribución de la carga mental para evitar sobrecarga y especializar el conocimiento, la ejecución en paralelo para ganar eficiencia, y la combinación de especializaciones para lograr una generalización robusta [35]. El resultado es una capacidad colectiva superior que supera las limitaciones fundamentales del agente único, haciendo viable la aplicación de LLMs a problemas de una complejidad previamente inalcanzable

2.5.1 Fundamentos de arquitecturas multiagente

Las arquitecturas multiagente definen cómo los agentes interactúan, perciben el entorno y toman decisiones dentro de un MAS, organizando el comportamiento colectivo mediante principios como autonomía, reactividad y proactividad, que permiten actuar de forma independiente, responder a cambios y anticiparse a las necesidades del sistema. Con base en estos principios, las arquitecturas pueden ser centralizadas, descentralizadas o híbridas, según el grado de control y la distribución de tareas, adaptándose al tipo de problema y al entorno; todo ello se articula mediante el ciclo de percibir, razonar y actuar, que resume el funcionamiento esencial de un agente y garantiza una conducta dinámica y adecuada frente a las condiciones del sistema.

- **Fundamentos de los Agentes de IA**

  Un agente de IA es una entidad de software autónomo que percibe su entorno, toma decisiones y ejecuta acciones para cumplir un objetivo. El esquema tradicional de percibir, razonar y actuar corresponde directamente con la arquitectura de los agentes basados en LLM, donde los módulos de Percepción, Cerebro y Acción representan la aplicación práctica de estas funciones fundamentales [36], [37].

  - **Percibir/Percepción:** Representa la etapa de captura de información. Esta interfaz multimodal procesa datos provenientes del entorno, ya sean textos, imágenes, audio o estados internos del sistema, y los transforma en señales comprensibles para el componente cognitivo [36], [37].
  - **Razonar/Cerebro:** El Cerebro, basado en un modelo de lenguaje con memoria de corto y largo plazo, inicia la fase de razonamiento, combinando los estímulos percibidos con su conocimiento acumulado para generar interpretaciones, planificar estrategias o decidir acción [36], [37].
  - **Actuar/Acción:** Representa la etapa final en la que las decisiones se traducen en operaciones concretas, como generar texto, invocar APIs, manipular sistemas o interacción con el entorno físico. En sistemas multiagente, este componente habilita la coordinación entre agentes y con actores humanos [36], [37].

La configuración Percepción–Cerebro–Acción puede verse como una versión más actual del clásico ciclo de Percibir–Razonar–Actuar. En esencia, esto permite que los agentes basados en LLMs funcionen como unidades tanto de estudio como de diseño de algoritmos operativos dentro de un contexto multiagente, uniendo, de alguna forma, la teoría clásica de la IA con las capacidades computacionales más avanzadas.

- **Características fundamentales de los Agentes de IA**

  Los MAS representan un paradigma de la IA distribuida, compuesto por entidades inteligentes llamadas agentes que interactúan para resolver problemas que superan las capacidades individuales. Un agente se caracteriza por cuatro propiedades fundamentales, mostradas en la Figura 4, que definen su arquitectura dentro de un MAS. A continuación, se explica cada una de estas propiedades.

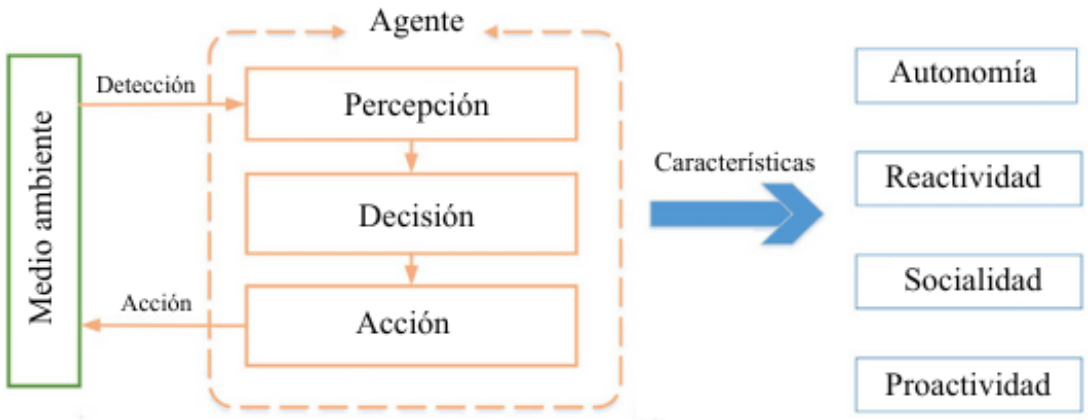


Figura 4.
Características fundamentales de un agente (adaptado de [38])

- **Autonomía:** Capacidad del agente para operar de forma independiente, sin intervención externa constante, y controlar sus acciones internas y su toma de decisiones [38].
- **Reactividad:** Habilidad del agente de percibir y responder dinámicamente a los cambios del entorno, traduciendo la información sensorial en acciones concretas y oportunas [38].
- **Proactividad:** Cualidad que le permite al agente no solo reaccionar a cambios externos, sino también tomar la iniciativa y anticiparse a escenarios futuros [38].

- **Socialidad:** Capacidad del agente para comunicarse e interactuar con otros, permitiendo la coordinación y cooperación necesarias para enfrentar tareas complejas que no podrían abordarse de forma individual [38].

Estos fundamentos son la base sobre la que se diseñan y construyen las arquitecturas multiagente. A partir de ellos es posible desarrollar sistemas inteligentes y flexibles, capaces de operar en entornos complejos, colaborar entre sí y adaptarse a condiciones cambiantes.

- **Arquitecturas de comunicación para LLM-MAS**

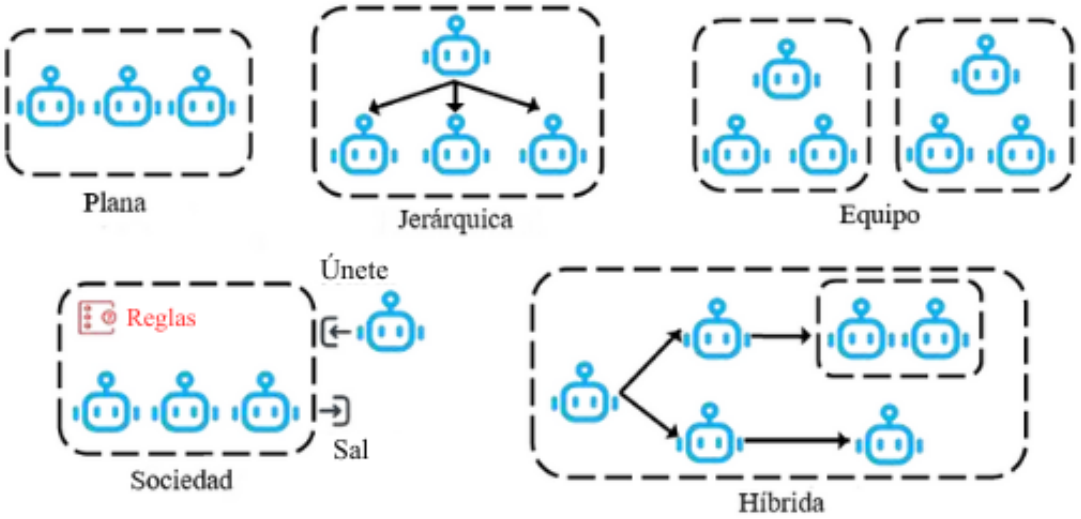


Figura 5.
Arquitecturas de comunicación para LLM-MAS (adaptado de [37])

La arquitectura de comunicación define cómo se organizan los agentes en un sistema multiagente basado en LLM y cómo circula la información entre ellos. Este diseño influye en el comportamiento global, las interacciones entre agentes y atributos como escalabilidad, adaptabilidad y rendimiento. Su elección depende de la complejidad de la tarea, el grado de autonomía de los agentes y el tipo de interacción que se busca fomentar. Como se muestra en la Figura 5, se consideran cinco configuraciones principales de comunicación [37]:

- **Arquitectura Plana:** Basada en una red descentralizada donde los agentes interactúan como pares, sin jerarquías ni un controlador central. Es útil en contextos que

requieren interacciones rápidas, flexibles y no estructuradas, como reasignaciones dinámicas de tareas o decisiones inmediatas. Aunque favorece la adaptabilidad, puede presentar dificultades al escalar, pues el aumento de agentes incrementa el costo de comunicación y reduce la eficiencia de la coordinación [37].

- **Arquitectura Jerárquica:** En este modelo, los agentes son jerárquicos: los que están en la parte superior asumen funciones más estratégicas, mientras que, por otro lado, hay agentes cuyo único trabajo consiste en realizar ciertas tareas que, en última instancia, alimentan toda la operación. Es adecuada para problemas complejos que requieren delegación clara, supervisión e integración de distintas competencias. No obstante, puede generar cuellos de botella cuando los agentes de alto nivel se saturan o surgen demoras en la comunicación entre capas [36], [37].
- **Arquitectura de Equipo:** En este enfoque, los agentes se agrupan en equipos especializados orientados a tareas o dominios específicos, lo que permite aprovechar la experiencia de cada grupo y fortalecer la cooperación interna. Si bien favorece al máximo la colaboración dentro de cada equipo, las interacciones entre grupos suelen incrementar la carga de comunicación, por lo que se requieren mecanismos eficientes de coordinación para mantener un desempeño óptimo [37].
- **Arquitectura de la Sociedad:** Este tipo de arquitectura modela entornos sociales complejos en los que los agentes interactúan siguiendo reglas, normas o convenciones compartidas, reflejando dinámicas similares a las observadas en las sociedades humanas. En consecuencia, resulta especialmente adecuada para sistemas de gran escala que involucran agentes heterogéneos, roles diferenciados y múltiples motivaciones. Además, permite representar interacciones ricas y la aparición de comportamientos emergentes, lo que favorece la simulación de escenarios complejos. Sin embargo, esta misma riqueza de interacciones incrementa la complejidad del sistema, por lo que la gestión y el control global pueden volverse más difíciles de implementar y mantener [37].
- **Arquitectura Híbrida:** Muchos sistemas no se ajustan por completo a una única estructura, por lo que adoptan combinaciones de distintas arquitecturas. Los modelos

híbridos integran características de varios enfoques, por ejemplo, jerarquías combinadas con interacciones descentralizadas para construir sistemas más adaptables y capaces de enfrentar una amplia variedad de desafíos. Este diseño permite optimizar de forma dinámica la coordinación y la distribución de recursos, aunque requiere estrategias de implementación cuidadosas para evitar sobrecargas de gestión y mantener la coherencia operativa [37].

La arquitectura de comunicación es un elemento clave para el rendimiento, la adaptabilidad y la escalabilidad de los sistemas multiagente basados en LLM. Por ello, se debe evaluar con cuidado los requisitos de su solución, considerando los compromisos entre flexibilidad, control, escalabilidad y eficiencia con el fin de seleccionar la arquitectura o la combinación de arquitecturas más adecuada.

2.5.2 Roles y especialización de agentes

Los sistemas multiagente basados en LLMs ejecutan tareas complejas o representan escenarios sofisticados mediante la adopción de distintos roles. Según Xinyi Li et al.[36], la definición de estos roles requiere elaborar cuidadosamente de los perfiles de los agentes para asegurar que cada uno cumpla su función asignada. El mecanismo central de especialización es el perfil del agente, diseñado para crear entidades autónomas con estilos personalizados, permitiendo ejecutar sub-tareas específicas y definir su área de especialización dentro del equipo. Esta especialización implica que un agente asuma un único rol a la vez, evitando comportamientos incoherentes o interferencias en la coordinación.

Los roles representan funciones especializadas con responsabilidades concretas. Para Mohamed A et al.[13], estos roles permiten tomar decisiones autónomas, delegar tareas y contribuir coordinadamente al logro de objetivos comunes, favoreciendo la cooperación, la gestión eficiente y la integración de herramientas externas.

- **Agente Coordinador**

El rol del agente coordinador (u orquestador) en una arquitectura multiagente consiste en gestionar y orquestar las interacciones entre los distintos agentes del sistema. Supervisa la ejecución general, simplificando la toma de decisiones, evita conflictos y mejora la eficiencia global [36]. Sin embargo, la dependencia estructural en este agente central puede generar un punto único de fallo y retrasos en la comunicación, afectando la capacidad de respuesta del sistema.

El coordinador asigna dinámicamente las tareas según las capacidades y la carga de trabajo de cada agente, optimizando los recursos disponibles [36], y aplicando mecanismos para resolver conflictos de coordinación mediante métricas como la frecuencia de conflictos y el tiempo de resolución, manteniendo la eficiencia operativa.

- **Agente Extractor**

  El agente extractor es el componente encargado del ingreso y preprocesamiento de la información. Su función es recibir documentos de distintos formatos, como PDF o texto plano, y transformarlos en una representación estructurada y legible para otros agentes.

  Según Petros R et al. [5], este proceso combina técnicas de reconocimiento óptico de caracteres (OCR) con el análisis del diseño visual para extraer el contenido textual y conservar su organización básica: encabezados, párrafos y numeración de cláusulas. De este modo, el agente preserva la estructura original del documento, algo fundamental para mantener la coherencia semántica en las etapas posteriores de análisis.

- **Agente Analizador**

  El agente analista funciona como el centro de inteligencia del sistema, encargado de interpretar las consultas del usuario, anticipar necesidades implícitas, verificar la suficiencia de los datos obtenidos y consolidar información de múltiples fuentes en respuestas coherentes [39]. Evalúa la calidad y relevancia de la información extraída, verificando que las tareas estén dentro de sus capacidades. También aprende de la retroalimentación del coordinador y de experiencias previas, lo que le permite afinar su desempeño y cometer menos errores al identificar cláusulas.

2.5.3 Aplicaciones en procesamiento de documentos legales

En los últimos años, la IA ha ido transformando el ámbito jurídico al facilitar, cada vez más, el procesamiento y la gestión de documentos legales. Los modelos de lenguaje y las arquitecturas multiagente han hecho posible automatizar tareas que antes dependían casi por completo del análisis humano, lo que se traduce en mejoras claras en eficiencia, precisión y también en trazabilidad.

El resultado, al final, son sistemas más accesibles y comprensibles que permiten ver con mayor claridad el alcance real que puede tener el procesamiento automatizado dentro del campo legal.

En su trabajo, Petros R et al. [5], presentan PAKTON, un marco de IA de código abierto que automatiza el análisis de contratos legales mediante una arquitectura multiagente compuesta por un Archivista, un Investigador y un Interrogador, encargados de extraer, buscar y generar informes explicativos. Este sistema actúa como un asistente legal capaz de responder preguntas, identificar cláusulas, verificar coherencias y producir reportes basados en evidencias.

Todo esto refleja una integración cada vez más marcada entre las arquitecturas multiagente y el razonamiento colaborativo dentro del ámbito legal, que va, poco a poco, desde la extracción y análisis de contratos hasta la generación y evaluación de documentos más complejos.

2.6 Protocolo A2A

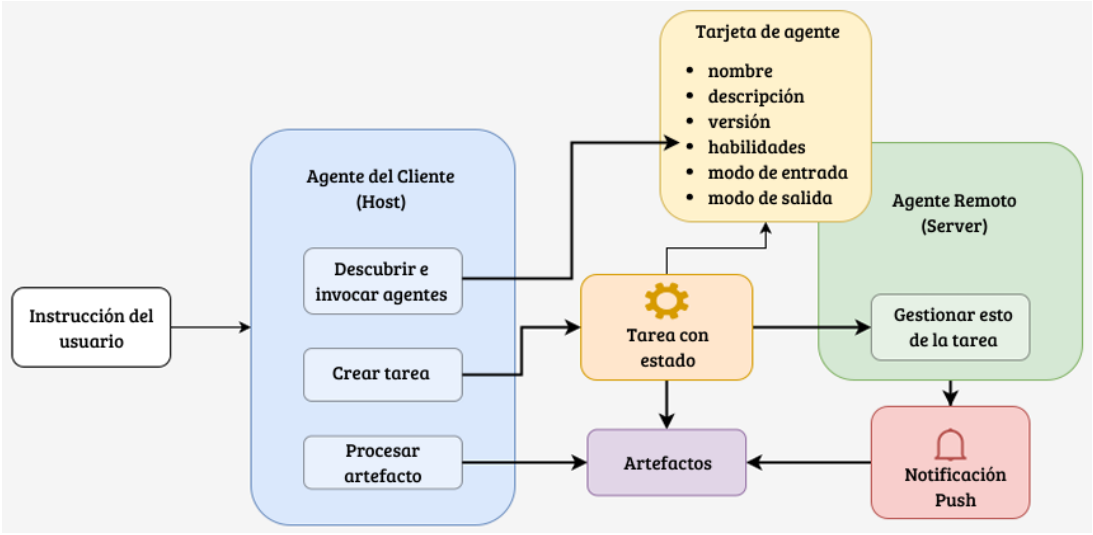


Figura 6.
Visión general del protocolo A2A (adaptado de [40])

El protocolo A2A, desarrollado por Google, representa un avance clave para establecer una comunicación estructurada, segura e interoperable entre agentes autónomos. A2A permite que los agentes se identifiquen mediante Tarjetas de Agente estandarizadas, utilicen métodos criptográficos para autenticarse y compartan tareas de forma declarativa y verificable. Su arquitectura responde a la creciente necesidad de sistemas de IA modulares capaces de operar y escalar entre distintos entornos, manteniendo flexibilidad ante flujos gestionados tanto por humanos como por otros agentes [41].

La llegada de A2A es muy relevante en un contexto donde la inteligencia artificial se orienta hacia ecosistemas más abiertos y basados en agentes, con aplicaciones en distintas áreas. A2A proporciona la base para construir entornos multiagente confiables, en los que la interoperabilidad y la seguridad son componentes esenciales de la infraestructura, y no simples complementos añadidos después [41].

2.6.1 Actores del Protocolo A2A

El Usuario inicia una solicitud sin tener que interactuar directamente con la infraestructura de agentes como se observa en la Figura 6. El Agente Cliente recibe dicha petición, interpreta su intención y selecciona el Agente Remoto adecuado revisando las capacidades descritas en su Tarjeta de Agente [40].

Una vez elegido, el Agente Cliente establece la comunicación con el Agente Remoto para ejecutar la tarea, gestionando el flujo de mensajes y recopilando los resultados producidos denominados Artefactos, que finalmente son entregados al Usuario [40]. A continuación, se describen en detalle estos tres actores:

- **Usuario**

  Es quien desencadena cualquier interacción en el protocolo A2A, pues representa la intención o necesidad que pone en marcha el funcionamiento del agente. Aunque suele ser una persona, también puede ser un sistema, un servicio o incluso otro agente. El Usuario no se comunica directamente con los Agentes Remotos, como lo indica la Figura 6; toda interacción se realiza a través del Agente Cliente, que traduce sus solicitudes en tareas ejecutables y administra las respuestas [40].

  A2A contempla diversos modos de interacción del usuario. Un **Usuario Final Directo** emplea interfaces como chatbots o asistentes de voz para enviar peticiones y recibir resultados. Un **Usuario Final Indirecto** utiliza aplicaciones o sistemas que integran agentes A2A sin que el usuario lo perciba. De igual forma, sistemas o servicios y agentes pueden actuar como Usuarios invocando acciones de otros agentes para resolver tareas más complejas [40].

- **Agente del Cliente**

  Actúa como intermediario al interpretar la intención del usuario y coordinar con los agentes remotos la ejecución de la tarea solicitada. Su funcionamiento abarca el ciclo de vida de la tarea, iniciando con el descubrimiento de agentes, donde consulta sus tarjetas para

identificar capacidades, formatos de entrada y salida y requisitos de autenticación, como se muestra en la Figura 6. Con esta información, selecciona el agente remoto más adecuado y procede a la iniciación de la tarea, generando una representación estructurada de la intención del usuario, los metadatos relevantes y los insumos formateados, que envía mediante un mensaje conforme al protocolo. Durante la ejecución, gestiona el intercambio de mensajes y artefactos, manteniendo comunicación bidireccional y preservando el contexto de sesión en tareas prolongadas o con estado [40].

Y además, el manejador también maneja errores: interpreta fallos del agente remoto y aplica mecanismos de recuperación; además de manejar la presentación de los resultados, transformando los Artefactos entregados en un formato que sirve al usuario. Cuando está habilitado, también maneja la comunicación assincronada por medio de SSE (Event Stream Server) y/o notificaciónes ”push”. Un Agente del cliente funciona también como orquestador, traductaor, y enlace de información entre el cliente y nosotros a la pregunta: en A2A, haciendo posibles interacciones inteligentes y contextuales con respuesta a la intención del usuario.

- **Agente Remoto (Servidor)**

  Actúa como el componente servidor encargado de ejecutar las tareas delegadas por el agente cliente, como lo indica la Figura 6. Ofrece una o varias habilidades, que pueden abarcar desde consultas simples hasta procesos más complejos que integran APIs externas o bases de datos. Cada habilidad dispone de un esquema formal de entrada y salida, lo que garantiza una invocación coherente por parte de cualquier cliente [40].

  Para exponer estas capacidades, el agente remoto publica su tarjeta de agente donde detalla sus habilidades, reglas de uso, formatos de datos, protocolos soportados y requisitos de autenticación. Esta tarjeta funciona tanto como anuncio público para otros agentes [40].

  Además de realizar tareas, puede gestionar sus propios recursos, así como también implementa mecanismos de seguridad y control de acceso, que incluyen autenticar a los clientes, verificar la integridad de los mensajes y autorizar el uso de habilidades según las políticas definidas [40].

2.6.2 Componentes Principales de A2A

Un agente A2A se organiza a partir de un conjunto de componentes que determinan su comportamiento, capacidades y forma de interactuar, como lo señala la Figura 6. Estos elementos constituyen la base operativa y semántica que permite a cualquier agente integrarse y funcionar dentro del sistema de agente a agente.

- **Tarjeta de Agente (Agent Card):** Es el mecanismo de autodescripción y descubrimiento. Se publica como un documento JSON que expone información como nombre, versión, descripción, habilidades y requisitos de autenticación. Sin una Tarjeta de Agente, un agente se vuelve invisible dentro del entorno A2A [40].

- **Habilidades (Skills):** Representan las funciones que un agente puede ejecutar. Cada habilidad detalla su nombre, propósito, parámetros de entrada y tipo de salida. Se activan mediante tareas y encapsulan la utilidad principal que el agente ofrece. La especialización y valor de un agente dependen directamente del alcance y precisión de sus habilidades publicadas [40].

- **Tarea (Task):** Es la unidad básica de delegación. Define qué habilidad debe ejecutarse, junto con sus parámetros e información contextual, las cuales son emitidas por agentes clientes y resueltas por agentes remotos, permitiendo tanto colaboración sincrónica como asincrónica. Al estandarizar la intención y la invocación, las tareas garantizan interoperabilidad entre agentes A2A en distintos entornos [40].

- **Mensajes (Messages):** Constituyen el canal central de comunicación entre agentes. Contienen el intercambio de datos y las acciones de coordinación: envío de tareas, actualizaciones de estado o entrega de artefactos. Pueden incluir texto plano, datos estructurados o referencias a archivos. Sin mensajes, la interacción entre agentes no sería posible, por lo que son un elemento esencial del protocolo [40].

- **Artefactos (Artifacts):** Son los resultados que produce el agente remoto al completar una tarea. Pueden tomar la forma de respuestas estructuradas, cálculos, documentos o datos

asociados, y se devuelven al agente cliente para que los presente al usuario o los use en pasos posteriores [40].

2.6.3 Capa de transporte A2A y Comunicación

El protocolo A2A soporta diversos mecanismos de transporte para la comunicación entre el cliente y los agentes remotos. Cuando se requiere transmisión en tiempo real y ambas partes lo permiten, se emplea SSE, que mantienen una conexión HTTP persistente mediante la cual el agente remoto envía actualizaciones o artefactos al agente cliente, proporcionando retroalimentación continua durante tareas prolongadas [40].

En escenarios donde las conexiones persistentes no son viables, el protocolo admite notificaciones push a través de la interfaz PushNotificationService, permitiendo informar sobre el avance o la finalización de tareas mediante canales externos, lo cual resulta adecuado para procesos tolerantes a la latencia y ejecuciones en segundo plano [40].

Las comunicaciones principales del flujo de tareas en A2A siguen la especificación JSON-RPC 2.0, garantizando un formato uniforme para la invocación de métodos, el envío de parámetros y la encapsulación de resultados. Asimismo, el descubrimiento de agentes remotos se realiza mediante solicitudes HTTP GET a sus endpoints, desde donde se obtiene la tarjeta de agente. En conjunto, estos mecanismos proporcionan una comunicación flexible, interoperable y ampliable en distintos entornos de ejecución [40].

2.6.4 Ciclo de vida del Agente Remoto (Servidor)

El ciclo de vida de un agente remoto en el protocolo A2A se organiza en cuatro fases: Creación, Operación, Actualización y Terminación, cada una orientada a garantizar un comportamiento seguro, detectable y confiable. La Creación inicia con la publicación de la tarjeta del agente en el endpoint /.well-known/agent.json, donde se exponen datos como nombre, versión, habilidades y métodos de autenticación [40].

A continuación, el servicio se habilita para procesar solicitudes JSON-RPC 2.0 sobre HTTP e implementa los mecanismos de autenticación declarados, asegurando la validación del cliente y el control de acceso [40].

Durante la Operación, el agente remoto atiende las tareas enviadas por los agentes clientes, ejecuta la lógica correspondiente y gestiona la comunicación mediante mensajes de estado y entrega de artefactos vía JSON-RPC. Cuando está habilitado, puede emitir actualizaciones mediante SSE o emplear servicios de notificación push, manteniendo además el estado interno de las tareas para garantizar estabilidad y trazabilidad durante todo el proceso [40].

La Fase de Actualización abarca cualquier cambio en las capacidades y configuraciones del agente, ya sea la incorporación de nuevos modelos o versiones, la adición o retiro de skills, o incluso la aplicación de parches de seguridad. Todo esto, en la medida de lo posible, se comunica a los clientes para mantener cierta transparencia en el proceso [40].

Por otro lado, en la fase de Terminación, el agente cierra sus operaciones de forma ordenada: completa las tareas pendientes, finaliza los flujos activos de comunicación con el cliente y, finalmente, desregistra su tarjeta para liberar el recurso. Se evita que los puntos terminales queden expuestos. Un ciclo de vida bien definido como este fortalece la interoperabilidad, la seguridad y la confiabilidad del ecosistema A2A [40].

2.6.5 Estándares de comunicación

El Lenguaje de Comunicación de Agentes (ACL), desarrollado por la Fundación para Agentes Físicos Inteligentes (FIPA), es un estándar que permite la comunicación coherente entre agentes de software autónomos al definir la sintaxis y semántica de los mensajes intercambiados, garantizando una comprensión mutua [42].

Además de representar el contenido informativo, los ACL incorporan la expresión de intenciones y expectativas, lo que posibilita interacciones estructuradas y orientadas a objetivos, así como la coordinación de acciones como el intercambio de conocimiento y la negociación

de tareas. En conjunto, constituyen un pilar fundamental en los sistemas multiagente, al permitir una comunicación autónoma, estructurada y con propósito para la resolución de problemas complejos [42].

- **Componentes de los ACL**

  Los ACL constituyen la base esencial para la interacción entre sistemas de software autónomos, desempeñando un papel similar al de los lenguajes humanos en los procesos de comunicación y colaboración [42].

  Todos estos lenguajes están construidos alrededor de unos componentes comunes formando una unidad integral: sintaxis, semántica y pragmática. Así pues, la sintaxis define a qué forma y formato obedecen los mensajes; la semántica establece cómo es necesario que signifiquen y cómo mantener una interpretación coherente; y la pragmática dice cómo usar mensajes según un contexto dado [42].

  Todos los mensajes en ACL contienen un grupo claves de campo: el performativo (que indica la intención comunicativa), el emisor y el receptor (los agentes involucrados en la comunicación), el contenido del mensaje, el ID de conversación (para mantener la continuidad del diálogo) y las definiciones de ontología y lenguaje para asegurar la misma interpretación entre todas las partes [42].

- **Idiomas de comunicación de agentes**

  La comunicación entre agentes autónomos es clave para que puedan coordinarse. En ese marco, dos lenguajes ocupan un lugar central en la interacción entre agentes: KQML y FIPA-ACL. Ambos funcionan como protocolos que facilitan el intercambio de información y la coordinación de acciones dentro de sistemas distribuidos.

  El Knowledge Query and Manipulation Language (KQML), desarrollado por Genesereth y Ketchpel, fue uno de los primeros marcos de comunicación entre agentes. Se basa en un modelo estructurado en mensajes organizados en capas funcionales e introdujo performativas o tipos de mensajes como ask-if, tell y reply junto con parámetros como, content, :language, :ontology y :receiver, lo que permitió una interacción más ordenada y comprensible entre las entidades participantes [40].

En seguida FIPA-ACL emergió como una evolución de KQML, poseyendo una estructura semántica más estricta basada en la teoría de los actos de habla. Con este abordaje los mensajes son entendidos como acciones con intenciones y efectos bien definidos; agregando exactitud y verificabilidad a la comunicación [42].

También expandió la colección de performativas (accept, reject y request) estandarizando las lenguas de contenido y estableciendo protocolos interactivos para diferentes modelos conversacionales [40]. Su implementación sobre plataformas tales como JADE y JACK fueron notables. Sin embargo, la complejidad del framework limitó su adopción.

En conjunto, KQML y FIPA-ACL crean las bases lingüísticas para una colaboración confiable entre agentes dentro de un entorno distribuido, usándolos como formatos estandarizados y, al mismo tiempo, permitiendo trabajar con distintos lenguajes de contenido y ontologías.

- **Ontología en la comunicación entre agentes**

  Una ontología constituye la base de la comunicación efectiva entre agentes al proporcionar un lenguaje común y una comprensión compartida del dominio en el que operan, permitiendo no solo el intercambio de mensajes, sino también la interpretación coherente de conceptos y relaciones [42].

  Estas definen un vocabulario formal, junto con las relaciones, reglas y restricciones asociadas, lo que posibilita que los agentes razonen sobre el entorno, actúen de manera coordinada y mantengan la consistencia semántica en el procesamiento de la información, facilitando la interoperabilidad entre sistemas heterogéneos [42].

  Las ontologías aseguran una interpretación precisa de los mensajes, incluso en entornos abiertos donde participan agentes desarrollados por distintas organizaciones. Son, en definitiva, un elemento sin el que la cooperación y el entendimiento mutuo entre agentes difícilmente serían posibles.

- **Evolución y direcciones futuras en la comunicación entre Agentes**

  La comunicación entre los agentes está atravesando un momento de cambio por las nuevas técnicas avanzadas de aprendizaje automatico y LLMs redefiniendo como los agentes

interactuan y colaboran en entornos complejos [42]. Ligado a esto, el progreso en la interacción humano-agente ha dado lugar a una apreciación más contextualizada y flexible de los mensajes, conduciendo así a una mejor coordinación para tareas distribuidas.

Así que hoy cobran protagonismo protocolos ligeros y estandarizados tales como MCP, ACP, A2A, que mejoran la interoperebilidad/scalabilidad cuando habilitan comunicación segura, descubierto dinámicamente y colaborativamente descentralizado en sistemas híbridos, de particular útilidad para entornos críticos. Hay que añadir a eso el hecho de que estos protocolos usan encriptación y autentificación avanzada que aumenta la confianza en escenarios muy delicados por naturaleza [40].

Por todo, existen desafíos reales: Desacuerdos semánticos entre agentes, ausencia de estandarización, una elevada carga computacional en modelos lingüísticos extensos, y problemas en el mantenimiento de la coherencia comunicativa en sistemas de múltiples agentes. Los obstáculos que resolveremos serán la clave hacia el logro de comunicación verdaderamente interoperable, escalable y segura en los sistemas multiagentes futuros.

2.6.6 Mecanismos de coordinación

Los mecanismos de coordinación constituyen, en el fondo, el núcleo de la interacción eficiente entre agentes en los sistemas multiagente, ya que permiten alcanzar objetivos colectivos sin depender de un control centralizado. A su vez, estrategias como la planificación colaborativa facilitan la sincronización de acciones y el intercambio de información y, por otro lado, la negociación y las subastas funcionan como medios bastante efectivos para resolver conflictos y asignar recursos de manera más equitativa.

En conjunto, estos enfoques han mostrado resultados sólidos en la mejora de la interoperabilidad, la coordinación y la eficiencia de sistemas multiagente complejos. De este modo, se confirma su relevancia en la construcción de entornos cooperativos y adaptativos, capaces de responder de manera más efectiva a los desafíos asociados con escenarios dinámicos y distribuidos.

- **Paradigmas de comunicación**



Figura 7.
Tipos de paradigmas de comunicación (adaptado de [37])

Los paradigmas de comunicación definen como los agentes representan, comparten e interpretan la información dentro de sistemas multiagente impulsados por modelos de lenguaje. Según Bingyu Yan et al [37], se destacan el uso de tres paradigmas mostrados en la Figura 7. Gracias a las capacidades avanzadas de comprensión y generación del lenguaje natural que ofrecen los LLM, estos enfoques permiten interacciones más contextuales y expresivas que los métodos simbólicos tradicionales.

- **Paso de Mensajes**

  El paradigma de Paso de Mensajes describe un tipo de comunicación directa, ya sea punto a punto o mediante difusión, en el que los agentes intercambian mensajes claros con información, instrucciones o peticiones. Estos mensajes suelen incluir contenido en lenguaje natural junto con datos contextuales y elementos de razonamiento, lo que permite una transmisión más precisa y efectiva de la información entre los agentes [37].

- **Acto de habla**

  El paradigma del Acto de Habla sostiene que la comunicación entre agentes no solo transmite información, sino que también ejecuta acciones que influyen en el comportamiento y el estado del sistema. Mediante enunciados con funciones directivas, persuasivas o comisivas, los agentes pueden coordinar tareas, negociar compromisos y adaptarse a contextos cambiantes. Sin embargo, su efectividad depende de la claridad semántica y la coherencia contextual, ya que ambigüedades o interpretaciones erróneas pueden generar fallos en la interacción [37].

- **Pizarra**

  El paradigma de la Pizarra se basa en un espacio centralizado donde los agentes comparten y actualizan información de forma colaborativa, lo que les permite coordinarse eficientemente y mantener una comprensión común del estado del sistema. Este enfoque es especialmente útil en entornos que exigen alta coordinación, como la toma de decisiones conjunta o la resolución distribuida de problemas, ya que facilita la alineación de estrategias y acciones [37].

- **Mecanismo de coordinación**

  - **Negociación**

    La negociación en sistemas multiagente es un proceso de interacción mediante el cual los agentes buscan alcanzar acuerdos mutuamente beneficiosos al gestionar recursos o resolver conflictos de intereses, reflejando comportamientos adaptativos y cooperativos que preservan la coherencia y eficiencia del sistema colectivo.

    Basada en el intercambio de actos de habla, permite a los agentes expresar ofertas, contraofertas, compromisos o rechazos, ajustando dinámicamente sus estrategias según la información recibida y utilizando elementos directivos, comisivos o persuasivos para coordinar acciones, ajustar creencias y optimizar decisiones durante la interacción [37].

    El uso de estrategias predefinidas y reglas de decisión facilitan la obtención autónoma de acuerdos equilibrados, mientras que la incorporación de técnicas de aprendizaje fortalece su capacidad de adaptación, aumentando la eficacia y precisión del proceso negociador [43], [44].

  - **Subastas**

    Las subastas en sistemas multiagente constituyen un mecanismo de coordinación que permite a los agentes autónomos competir o colaborar por recursos limitados mediante la formulación de ofertas basadas en sus preferencias o capacidades, mientras un mecanismo determina la asignación más eficiente, promoviendo la equidad, la descentralización y la coherencia con los principios de autonomía del sistema [45].

Asimismo, las subastas combinatorias facilitan la asignación eficiente de recursos interdependientes entre múltiples agentes con objetivos diversos, mediante reglas y algoritmos de oferta que posibilitan la negociación sobre conjuntos de recursos y la consecución de acuerdos sin un control centralizado; este enfoque ha sido fortalecido por avances en aprendizaje automático, blockchain y protocolos de comunicación como A2A, que mejoran su eficiencia, seguridad y capacidad de adaptación en tiempo real [45].

- **Planificación colaborativa**

  La planificación colaborativa o cooperación es una característica central de los sistemas multiagente, ya que permite que los agentes trabajen de forma conjunta hacia objetivos compartidos, combinando conocimientos y capacidades para lograr resultados que superan los esfuerzos individuales. Estas interacciones se sustentan en la transparencia, la confianza mutua y la orientación a metas comunes, promoviendo una coordinación efectiva y una relación sostenida entre los participantes [37].

  La cooperación entre agentes adopta dos formas principales. En la cooperación directa, los agentes intercambian información y recursos de forma fluida para resolver tareas juntos, priorizando la rapidez y la eficiencia. La cooperación mediante el debate, en cambio, introduce un análisis crítico de las aportaciones que permite cuestionar y refinar las propuestas hasta alcanzar conclusiones más sólidas [37].

2.6.7 Interoperabilidad entre frameworks

La interoperabilidad en los sistemas multiagente es la capacidad de distintos frameworks o plataformas para comunicarse e interactuar con eficacia, más allá de sus diferencias en lenguajes, protocolos o arquitecturas. Sin ella, construir entornos donde los agentes compartan información, coordinen acciones y alcancen objetivos comunes de forma autónoma sería considerablemente más difícil.

Entre los frameworks más destacados se encuentran:

- **JADE (Java Agent DEvelopment Framework)**, es un entorno, hecho en Java, para crear y manejar sistemas multiagente. Su compatibilidad con los estandares FIPA garantiza el intercanbio de información confiable entre agentes [46]. Esto permite usar esta tecnología como una opción fuerte para plataformas distribuidas que buscan flexibilidad e interoperabilidad.

- **SPADE (Smart Python multi-Agent Development Environment)**, escrito en Python, utiliza el protocolo XMPP/Jabber como base de comunicación. Aunque no implementa completamente las especificaciones FIPA, incorpora metadatos FIPA-ACL en su protocolo XML, lo que permite cierto nivel de interoperabilidad [47].

En cambio, la comunicación entre el JADE y SPADE es bastante limitada por las diferencias en los estándares de intercambio. JADE implementa completamente FIPA, mientras que SPADE solo lo hace en parte, lo cual dificulta su integración [47].

Con el fin de solucionar este problema, la interoperabilidad entre frameworks multiagente se basa cada vez más en los protocolos estandarizados (como el ACL) basados en los actos de habla definidos por FIPA y A2A, los cuales incluyen mecanismos para discovery autenticada y descubrimiento seguro.

Junto con enfoques recientes como MCP y ACP, estos protocolos permiten una comunicación más flexible y descentralizada, facilitando la integración de agentes heterogéneos en sistemas colaborativos modernos [37].

2.7 Extracción Automatizada de Cláusulas

La extracción automatizada de cláusulas se utiliza cada vez más en el análisis de documentos legales, especialmente por el aumento en la complejidad y el volumen de estos textos. Su propósito es identificar las secciones más relevantes de un contrato para facilitar su revisión y la evaluación de riesgos, tareas que tradicionalmente requieren la intervención de expertos [31]. En

este sentido, la automatización permite agilizar el proceso y mejorar el acceso a la información jurídica

A pesar de sus ventajas, los documentos legales presentan varios retos, como estructuras complejas, referencias cruzadas y variaciones en el significado de los términos, lo que dificulta su análisis con enfoques lineales [5]. Además, el carácter contextual del lenguaje jurídico complica el desempeño de los modelos tradicionales y de los enfoques monolíticos, que no siempre logran captar su organización ni su contexto.

Para abordar estas limitaciones, han surgido frameworks multiagente que utilizan protocolos de comunicación como A2A [6]. En este esquema, distintos agentes especializados colaboran durante el proceso de extracción. Por ejemplo, un agente extractor puede enviar cláusulas ambiguas, mediante mensajes JSON, a un agente encargado de validar la terminología jurídica, antes de que otro agente estructure los resultados en la respuesta final.

De esta manera, este tipo de coordinación hace posible mantener la coherencia a lo largo de todo el proceso y, al mismo tiempo, contribuye a mejorar la calidad de los resultados obtenidos en cada etapa.

Este enfoque, aplicado al dominio legal, facilitaría la creación de agentes especializados que, coordinados mediante estos protocolos, aseguren interoperabilidad, escalabilidad y trazabilidad en todo el proceso. La capacidad de estos sistemas para descomponer consultas complejas, enrutar subtareas a agentes específicos y sintetizar respuestas coherentes sugiere que es posible automatizar la extracción de cláusulas sin perder el contexto exhaustivo y la precisión exigidos por el análisis jurídico [6].

2.7.1 Técnicas de NLP para documentos legales

La aplicación del NLP en el ámbito legal, conocida como NLP jurídico, emplea técnicas computacionales para analizar, interpretar y automatizar tareas relacionadas con textos legales, permitiendo así optimizar el procesamiento y la gestión de información jurídica. En la práctica,

estas herramientas permiten transformar documentos no estructurados, como sentencias, leyes y contratos, en información más ordenada y procesable [48].

Gracias a esto, se pueden automatizar procesos como el resumen de documentos, la respuesta a preguntas o la clasificación de textos, lo que, poco a poco, va facilitando el trabajo con grandes volúmenes de información jurídica [48].

Los textos jurídicos a menudo resultan difíciles de procesar para los modelos de NLP debido al uso del lenguaje técnico, las estructuras complejas, la ambigüedad y las frecuentes referencias cruzadas. En consecuencia, tiene que aplicar enfoques propios del dominio legal a fin de extraer resultados adecuados.

- **NER jurídico**

  El Reconocimiento de Entidades Nombradas (NER) jurídico es una tarea especializada del NLP cuyo objetivo es identificar y clasificar menciones de texto en tipos predefinidos propios del dominio legal. Se enfoca en extraer entidades como leyes, normas, términos procesales, referencias a artículos, cortes, jueces y partes involucradas [48]. Esta identificación precisa es esencial para estructurar documentos legales y mejorar los sistemas de recuperación de información jurídica.

  Las técnicas de NER en el ámbito jurídico han pasado de enfoques basados en reglas y modelos estadísticos a métodos de aprendizaje profundo. En este contexto, Farid A et al. [48]indican que los enfoques actuales utilizan modelos tipo transformer, como BERT, los cuales suelen combinarse con capas Bi-LSTM y CRF. Además, incorporan embeddings y listas de entidades que mejoran la representación del texto y, en consecuencia, favorecen un mejor desempeño en el reconocimiento de entidades.

  Gracias a estas técnicas, es posible automatizar tareas como la generación de resúmenes, la extracción de cláusulas y la identificación de actores legales, lo que, en la práctica, facilita bastante el análisis de documentos jurídicos.

- **Segmentación Semántica**

  La segmentación semántica de documentos legales consiste en dividir el texto en unida-

des funcionales y asignarles roles específicos como Hechos, Problema Legal, Argumentos, Razonamiento o Decisión [49]. A través de este proceso, los documentos legales no estructurados se organizan en componentes funcionales, lo que permite comprender su estructura lógica y argumentativa más allá de su formato textual.

Para ello, se usan sobre todo modelos del aprendizaje profundo. Especialmente, arquitecturas transformer modificados para el contexto legal, además de técnicas de análisis semantico y clasificación de párrafos [48]. Estos sistemas usan mecanismos de atención para la representación más precisa del lenguaje legal y sus relaciones contextuales.

- **Integración de ambas técnicas**

  La segmentación semántica y el NER jurídico trabajan de forma complementaria para transformar un documento legal en información estructurada. En primer lugar, la segmentación semántica divide el texto en partes funcionales, como hechos, argumentos o cláusulas, permitiendo así comprender con mayor claridad su organización general. Posteriormente, el NER identifica dentro de esos segmentos las entidades clave, como personas, fechas, montos o normas. Como resultado de esta integración, se obtiene un análisis más preciso, contextualizado y útil para aplicaciones prácticas del ámbito jurídico [48].

  Un ejemplo de tal enfoque lo muestra Farid A. et al. [48], quienes describen su aplicación en el análisis automático de contratos. Para ilustrar esto, en este caso es lo que hará un modelo de segmentación semántica, primero identificando la estructura para poder dividirla por cláusulas. Un sistema de NER entonces extraerá las entidades relevantes: las partes, los montos, las fechas y referencias legales. A partir de esta información, se pueden generar resúmenes estructurales sobre derechos y deberes con contratos complejos.

2.7.2 Almacenamiento y recuperación de cláusulas

El almacenamiento y la recuperación de información en textos jurídicos siguen una estructura jerárquica donde artículos, párrafos y cláusulas actúan como unidades semánticas autónomas que pueden procesarse de forma independiente para mejorar la precisión en la búsqueda.

En este contexto, una cláusula contractual se considera una unidad de conocimiento, ya que encapsula un significado jurídico completo, permite un análisis aislado y facilita que los modelos de recuperación identifiquen fragmentos relevantes sin depender del documento completo [50].

- **Uso de Chunking y Embeddings en documentos legales**

  El chunking de documentos es una técnica de preprocesamiento que divide un texto en fragmentos más pequeños y coherentes, útil especialmente en el ámbito legal, donde cada cláusula o artículo tiene un significado propio. En documentos estructurados como contratos, segmentar por secciones o cláusulas garantiza que cada fragmento represente una unidad semántica completa, facilitando la posterior recuperación de información y análisis de contenido [3], [50].

  Los embeddings son representaciones vectoriales densas generadas a partir de estos fragmentos mediante modelos de lenguaje, que capturan el significado contextual, relaciones conceptuales y matices lingüísticos. Esta representación permite comparar textos de manera semántica, convirtiendo documentos y consultas en vectores que reflejan similitud, y almacenándolos en bases de datos vectoriales para búsquedas más precisas y eficientes que los métodos basados en palabras clave [3], [50].

  En el ámbito jurídico, los embeddings permiten recuperar información en función de su significado, mantener el contexto legal y, además, analizar documentos complejos de una forma más eficiente y práctica. Facilitan la identificación de patrones entre contratos, la comparación de cláusulas y el ajuste de resultados mediante metadatos, reemplazando búsquedas literales por una comprensión semántica profunda, lo que los convierte en una herramienta esencial para gestionar información legal en dominios complejos [3], [50].

- **Bases de datos tradicionales y vectoriales**

  Las bases de datos tradicionales almacenan documentos estructurados y no estructurados, recuperando información mediante tablas relacionales o índices invertidos basados en coincidencias literales de términos exactos o parciales. Si bien estos mecanismos son eficientes para localizar información específica, su funcionamiento se fundamenta principalmente en la comparación directa de palabras y patrones textuales [51].

No obstante, este enfoque presenta una limitación importante: no es capaz de interpretar variaciones semánticas. En consecuencia, en el ámbito jurídico resulta más difícil identificar cláusulas redactadas mediante sinónimos o expresiones equivalentes. Esto se debe a que los documentos suelen registrarse mediante palabras clave y metadatos que no capturan relaciones conceptuales profundas. Por ello, el sistema compara los términos de forma literal, sin comprender su significado ni el contexto en el que aparecen, lo que restringe la recuperación de información relevante y dificulta la realización de análisis avanzados [51].

Por el contrario, una base de datos vectorial organiza y almacena datos estructurados o no estructurados junto con sus incrustaciones vectoriales, que representan dicho contenido en forma numérica. Esto permite localizar y recuperar rápidamente objetos similares a gran escala, razón por la cual también se conoce como motor de búsqueda vectorial. Su funcionamiento se basa en la búsqueda vectorial, que utiliza estas representaciones comprensibles por máquina para identificar similitudes entre elementos, junto con técnicas de indexación vectorial diseñadas para gestionar grandes volúmenes de información de manera eficiente [52]. En el ámbito jurídico, esto facilita encontrar cláusulas relacionadas con ciertos conceptos, incluso cuando están redactadas con sinónimos o expresiones equivalentes, proporcionando mayor eficiencia y precisión en la búsqueda legal.

En este contexto, existen diversas soluciones de código abierto orientadas a la gestión de vectores y búsquedas semánticas, entre las cuales destacan:

- **Weaviate**

  Es una base de datos vectorial de código abierto y nativa de la nube que almacena objetos y sus incrustaciones para habilitar búsquedas semánticas a gran escala. Integra en una sola interfaz la búsqueda por similitud, el filtrado por palabras clave, RAG y la reclasificación, lo que la vuelve bastante útil para motores de recomendación, chatbots y también para la búsqueda semántica tanto de texto como de imágenes. Por otro lado, ofrece opciones de despliegue mediante Docker o Kubernetes y cuenta con bibliotecas cliente para lenguajes como Python y Java [53], [54].

- **Chroma**

  Es una base de datos vectorial de código abierto enfocada en aplicaciones de IA, que facilita a los modelos de lenguaje el acceso a información relevante mediante el almacenamiento de incrustaciones y metadatos. Permite búsquedas vectoriales y de texto completo, gestión de documentos, filtrado por metadatos y recuperación multimodal. Además, se integra fácilmente en sistemas basados en LLM y ofrece soporte para herramientas como LangChain en Python y JavaScript [55], [56].

- **Milvus**

  Es una base de datos vectorial de código abierto, escalable y de alto rendimiento, diseñada para almacenar y buscar vectores derivados de datos no estructurados como texto, imágenes o audio. Permite búsquedas semánticas a gran escala y ofrece un modelo de datos flexible con soporte para tipos numéricos, vectores, matrices y JSON [57], [58].

  Puede desplegarse desde entornos locales con Milvus Lite hasta clústeres distribuidos en Kubernetes. Además, admite múltiples tipos de búsqueda, cuenta con APIs y SDK oficiales (Python, Java, Go, Node.js, entre otros) y se integra con herramientas de IA como LangChain [57], [58].

- **Qdrant**

  Se trata de una base de datos de vectores de tipo open source, desarrollada específicamente para almacenar y buscar vectores de alta dimensión en tareas de IA como búsqueda de similitudes, NLP y análisis de imágenes. Permite despliegues tanto locales como en la nube, admite distintos tipos de vectores y utiliza técnicas de cuantización para mejorar el rendimiento [59].

  Gracias a su naturaleza abierta y a su integración mediante API y SDK, facilita la implementación eficiente y de bajo costo de sistemas RAG y motores de recomendación. Además, ofrece un manejo ágil de vectores y metadatos en formato JSON, lo que la convierte en una opción accesible para quienes desean experimentar con la búsqueda vectorial [59].

- **Pipeline de construcción de la base de conocimiento**

El pipeline de construcción de una base de conocimiento transforma documentos complejos, como textos legales, en un repositorio inteligente basado en significados semánticos, permitiendo que sistemas de IA, como los basados en RAG, respondan consultas en lenguaje natural de manera rápida y contextualizada. El proceso consta de tres fases: en la indexación, los documentos se fragmentan en niveles jerárquicos y cada segmento se convierte en un embedding que captura su significado; en la recuperación, las preguntas del usuario se transforman en vectores y se comparan con la base para identificar los fragmentos más relevantes; y en la generación, un modelo de lenguaje sintetiza esos fragmentos para producir respuestas coherentes y precisas [50].

- **Búsqueda Semántica y Evaluación de Precisión en Documentos Legales**

  Las búsquedas por similitud son una técnica de recuperación de información que identifica contenido relevante comparando el significado semántico de los textos en lugar de depender de palabras clave exactas. Documentos y consultas se transforman en vectores numéricos (embeddings), y mediante métricas de similitud, como la distancia coseno, se recuperan los fragmentos más cercanos a la intención del usuario, lo que resulta útil en documentos complejos como contratos legales. La precisión de esta recuperación se evalúa mediante pruebas con usuarios especialistas y métricas como recall, MRR y precisión semántica, clasificando las respuestas según su corrección y completitud, lo que permite cuantificar el rendimiento del sistema y asegurar que la información recuperada sea correcta, relevante y útil [3].

2.7.3 Evaluación de precisión

La evaluación de precisión permite medir el rendimiento de los modelos de NLP aplicados a textos jurídicos, asegurando que las respuestas generadas sean realmente confiables y relevantes. Y es que, en este contexto, donde la exactitud es clave, se suelen utilizar tres métricas principales: precisión, recall y F1-score, que evalúan la calidad en la identificación y recuperación de cláusulas contractuales.

La precisión mide la proporción de resultados recuperados que son realmente relevantes, indicando cuántas cláusulas devueltas por el sistema son efectivamente pertinentes a la consulta, minimizando información irrelevante. El recall evalúa la cantidad de resultados relevantes recuperados del total disponible, mostrando qué tan completa es la búsqueda y asegurando que no se omitan cláusulas importantes [51].

El F1-Score es la media armónica entre precisión y recall, proporcionando un único valor que equilibra ambos aspectos, reflejando tanto la exactitud como la completitud de las respuestas, siendo especialmente relevante cuando falsos positivos o negativos pueden tener consecuencias significativas [51].

Un buen balance con respecto a estas métricas permite dar respuestas precisas sin perder información relevante pero de todos modos se ofrece resultados más completo y confiable.

2.8 Tecnologías de Implementación

Los marcos de agentes de IA permiten desarrollar sistemas inteligentes impulsados por LLM al facilitar la creación de agentes autónomos capaces de comprender lenguaje, razonar y tomar decisiones [13]. En los últimos años han surgido numerosos frameworks de código abierto, como LangChain, AutoGen y Haystack, que simplifican la construcción de agentes basados en LLM [60].

En la implementación de éstos sistemas se utilizan arquitecturas Web y herramientas de despliegue que permiten el escalado. El backend normalmente usa frameworks tipo Flask y FastAPI y los frontends utilizan cosas como React o Vue.js para la interfaz de usuario. Y hablando del despliegue, podemos desplegar con Docker, Kubernetes y los servicios en la nube (AWS/Azure) a los agentes de forma más flexible.

2.8.1 Frameworks para agentes

Los marcos de agentes de IA ofrecen un enfoque renovador para desarrollo sistemas inteligentes, integrando el poder de los LLM con herramientas modulares para crear agentes de software autónomos [13]. Proporcionan la arquitectura y las herramientas necesarias para diseñar, coordinar y desplegar agentes inteligentes [12]. Entre ellos destacan los siguientes:

- **LangChain**

  Es un marco de código abierto diseñado para simplificar el desarrollo de agentes al integrar LLM con distintas herramientas y fuentes de datos. Combina componentes preempaquetados, como modelos de lenguaje conversacionales, integraciones con motores de búsqueda y utilidades, en flujos que permiten razonamiento y toma de decisiones en múltiples pasos [13].

  Permite crear agentes personalizados definiendo roles, tareas y herramientas para que analicen un prompt, seleccionen la herramienta adecuada y refinen su respuesta [13].

- **AutoGen**

  Es un marco de código abierto que permite crear aplicaciones basadas en LLM mediante agentes que colaboran entre sí para completar tareas. Sus agentes son personalizables y pueden combinar LLM, intervención humana y herramientas externas. El desarrollador define cómo interactúan, usando tanto lenguaje natural o código para configurar sus patrones de conversación [61]. Funciona como un marco general para construir aplicaciones de distinta complejidad basadas en múltiples agentes.

- **Haystack**

  Es un marco de orquestación de IA de código abierto que permite construir aplicaciones basadas en LLM, desde agentes autónomos hasta sistemas RAG y soluciones multimodales listas para producción. Su diseño modular facilita combinar tecnologías de OpenAI, Google, Anthropic y proyectos como Transformers o Elasticsearch. Su núcleo se organiza en componentes y pipelines apoyados por document stores, agentes, herramientas e integraciones que simplifican el desarrollo de aplicaciones escalables [62].

- **CrewAI**

  Es un marco de código abierto el cual coordina equipos de agentes de IA para resolver tareas complejas de forma colaborativa. Su diseño ligero se organiza en cuatro componentes: la Tripulación (Crew), que orquesta el flujo de trabajo; los Agentes (Agents), especializados según su función; el Procesos (Process), que define los objetivos; y las Tareas (Tasks), que establecen los objetivos a cumplir [13], [63].

  Además, incorpora agentes basados en roles con memoria propia, lo que refuerza la coordinación, la delegación y la colaboración en entornos multiagente [12].

2.8.2 Plataformas web

Las plataformas web modernas combinan frameworks ligeros y eficientes en el backend, como Flask y FastAPI, con bibliotecas de frontend centradas en interfaces interactivas, como React y Vue. Esta integración permite desarrollar aplicaciones web escalables, modulares y de alto rendimiento, capaces de ofrecer experiencias dinámicas y una comunicación fluida entre cliente y servidor. Entre las que destacan las siguiente:

- **Flask**

  Es un framework web ligero, ideal para proyectos que requieren flexibilidad. Su diseño simple e intuitivo permite crear aplicaciones web y API de forma rápida, sin imponer una estructura rígida. Además, soporta múltiples extensiones como Flask-SQLAlchemy o Flask-WTF, lo que facilita agregar funciones según las necesidades. Por su naturaleza adaptable, se usa con frecuencia en prototipos y para proyectos que requieren control total del desarrollo [64].

- **FastAPI**

  Es un framework web moderno en Python. Gracias a su soporte para anotaciones de tipo, genera documentación compatible con OpenAPI. Su arquitectura ASGI y soporte asíncrono lo hacen rápido y adecuado para microservicios y aplicaciones de alto rendimiento,

como sistemas en tiempo real. Destaca por su modelo asincrónico, baja sobrecarga para crear API escalables y facilidad de uso para principiantes [64].

- **React**

  Uno de los frameworks de frontend más utilizados es React, que destaca por su sistema de componentes, el cual favorece la modularidad, la reutilización y el mantenimiento del código. Cada componente integra JSX, lógica en JavaScript y estilos en CSS, lo que permite construir interfaces escalables [65].

  A su vez incorpora una gestion de estado, un flujo de datos unidireccional y un modelo renderizado declarativo, haciendo simple el trato con los cambios dinamicos y actualizándolos de la interfaz de forma predecible.

- **Vue.js**

  Es un framework de JavaScript orientado a crear interfaces de usuario. Incorpora un enlace de datos reactivo que conecta los datos con el DOM y destaca por su tamaño ligero, que mejora el rendimiento en descarga y ejecución. Permite modificar fácilmente los componentes, es de código abierto y mantiene una sintaxis breve y clara, lo que facilita su uso incluso a desarrolladores principiantes [66].

2.8.3 Herramientas de despliegue

Herramientas modernas de despliegue tales como: Docker, kubernentes, servicio de nubes, etc. AWS y Azure ofrecen maneras de ejecutar las apps de manera eficienta, escalada y portable. Con el uso de contenedores, la orquestación automatizada y la infraestructura gestionada en la nube, estas tecnologías simplifican bastante el manejo de entornos complejos y, al mismo tiempo, aseguran despliegues más robustos y estables.

Las modernas herramientas de despliegue tal cual Docker, kubernetes y servicios en la nube de Amazon Web Servicios y Microsoft Azur proporcionan a los recursos de cómputo la facilidad de ejecutar y escalar. Gracias a los contenedores, la orquestación automatizada, la infrastructura en la nube, entre otros, se pueden manejar entornos complejos con mayor sencillez.

En este contexto, Docker permite crear y ejecutar aplicaciones dentro de contenedores que incluyen todas sus dependencias, garantizando así un comportamiento consistente en distintos entornos. Para ello, estos contenedores se generan a partir de imágenes que integran el código, las librerías y la configuración necesaria. Además, a diferencia de las máquinas virtuales, comparten el kernel del sistema host, por lo que resultan más ligeros y eficientes en términos de uso de recursos.

Por ello, Docker resulta especialmente útil en despliegues ágiles, arquitecturas basadas en microservicios y entornos de desarrollo reproducibles. Docker opera bajo una arquitectura cliente–servidor, en la que el cliente envía instrucciones al daemon encargado de construir y gestionar los contenedores, pudiendo comunicarse localmente o de forma remota mediante una API REST [67].

Kubernetes, es un sistema de software de código abierto orientado al despliegue, escalado y gestión automática de aplicaciones en contenedores, eliminando procesos manuales repetitivos y consolidándose como una de las herramientas más reconocidas en la gestión de contenedores por su adopción, eficiencia y versatilidad. Su arquitectura se organiza en un clúster compuesto por el plano de control y los nodos trabajadores: el plano de control integra componentes esenciales como kube-apiserver, kube-scheduler, kube-controller-manager, etcd y kubectl, mientras que los nodos trabajadores alojan las aplicaciones mediante kube-proxy, kubelet y pods, siendo estos últimos la unidad más pequeña del sistema, conformada por uno o más contenedores con su código y dependencias empaquetadas para ejecutarse en cualquier entorno informático [68].

De esta forma, Docker crear y ejecutar contenedores individuales, mientras que Kubernetes es un sistema de orquestación que gestiona, escala y automatiza el despliegue de múltiples contenedores en un cluster. Kubernetes complementa a Docker al ofrecer capacidades como autoescalado, balanceo de carga y recuperación automática de contenedores fallidos [69].

Amazon Web Services y Microsoft Azure son plataformas de computación en la nube que permiten desplegar y gestionar aplicaciones mediante servicios escalables y administrados. AWS destaca con Amazon SageMaker y las instancias EC2 con GPU, mientras que Azure sobresale

con Azure OpenAI Service y su integración empresarial [70]. En conjunto, proporcionan la base necesaria para atender las altas exigencias computacionales de la IA moderna.

Sobre esta base de servicio cloud, ambas plataformas ofrecen servicios nativos totalmente gestionados para esta pila de contenedores: AWS cuenta con Elastic Kubernetes Service (EKS) AmazonWebServicie2025 y Azure ofrece Azure Kubernetes Service (AKS) MicrosoftLearn2025. Esta integración permite desplegar, gestionar y escalar aplicaciones de manera rápida y confiable, sin la carga de mantener la infraestructura principal, consolidando a Docker y Kubernetes como componentes esenciales en el ecosistema de la nube [70].

2.9 Métricas de Evaluación para Agentes IA

Conforme los agentes de IA autónomos son empleados en aplicaciones del mundo real emerge la necesidad de evaluar el desempeño del trabajo más cabalmente. Esos sistemas operan sobre ambientes dinámicos y se relacionan mutuamente; no es suficiente simplemente medir el grado de precisión. Por otra parte, se deben tener en cuenta la coordinación, el nivel de las respuestas, y escalabilidad del sistema.

2.9.1 Eficiencia de coordinación

La coordinación en los MAS basados en LLM representa un desafío para el desempeño de sistemas complejos. La eficiencia de coordinación no solo depende de la calidad individual de cada agente, sino también de la efectividad de sus protocolos de comunicación y de la topología empleada [35].

- **Métricas de Tiempo de Respuesta**

  El tiempo de respuesta en sistemas multiagente puede descomponerse en tres dimensiones fundamentales, respaldadas por estudios recientes en arquitecturas basadas en LLM:

  - **Latencia de razonamiento:** Tiempo que un agente requiere para procesar infor-

mación, generar inferencias y producir una acción o decisión. En arquitecturas impulsadas por LLM, depende del tiempo de inferencia del modelo y del manejo del contexto acumulado [71].

- **Latencia de comunicación:** Tiempo asociado al intercambio de mensajes entre agentes. La coordinación distribuida indica que los mecanismos basados en comunicación por eventos pueden mejorar la eficiencia y la escalabilidad frente a comunicaciones periódicas [72], [73].
- **Tiempo de coordinación total:** Métrica global que captura el tiempo necesario para completar una tarea colaborativa desde su inicio hasta su finalización. Algunas propuestas introducen KPIs que miden el progreso, la contribución de cada agente y la eficiencia temporal del ciclo de coordinación [74].

- **Carga de Mensajes A2A**

  La carga de comunicación A2A constituye, en la práctica, un factor bastante crítico para la eficiencia del sistema, ya que puede influir directamente en su rendimiento general. La investigación propone tres métricas principales:

  - **Entropy Efficiency Index (IEI):** Este mide el índice de eficiencia en la representación de datos. Valores más altos significan que los mensajes son mejores e informativos [75].
  - **Specialization Efficiency Index (SEI):** Mide el grado de diferenciación funcional entre agentes mediante la diversidad de mensajes, indicando la existencia de roles especializados que mejoran a la coordinación efectiva [75].
  - **Topology Efficiency Index (TEI):** Cuantifica el éxito de las tareas en relación con la frecuencia de comunicación, midiendo el consumo de recursos, como los costes computacionales, el uso de ancho de banda y también el overhead temporal [75].

2.9.2 Calidad de respuestas

La calidad de las respuestas generadas por agentes de inteligencia artificial es, en la práctica, un aspecto clave para asegurar su utilidad en aplicaciones reales. Evaluarla implica analizar distintas dimensiones y, entre ellas, la exactitud semántica suele ser de las más importantes, ya que permite ver si el agente no solo produce respuestas coherentes, sino también correctas en cuanto a significado y fidelidad factual respecto a la información de entrada.

- **Exactitud Semántica**

  La exactitud semántica evalúa qué tan bien las respuestas generadas por agentes IA capturan el significado correcto y mantienen fidelidad factual con respecto a la información de entrada. Entre las métricas utilizadas se encuentran:

  - **BERTScore:** Evalúa la similitud semántica mediante embeddings contextuales generados por modelos preentrenados. Calcula la similitud coseno entre tokens para medir coincidencia semántica profunda, más allá de la similitud léxica. Es ideal para determinar la fidelidad del significado entre la respuesta y un texto de referencia [76].
  - **MoverScore:** Mide la similitud entre textos a partir de embeddings contextualizados, considerando coincidencias parciales entre palabras o n-gramas.Sus resultados muestran, en general, una buena correspondencia con evaluaciones humanas en distintas tareas, lo que refuerza su utilidad en la práctica [77], [78].
  - **Sentence Mover Similarity (SMS):** Métrica automática que evalúa la similitud entre textos extensos mediante embeddings de oraciones en lugar de palabras. Representa cada documento como un conjunto de oraciones vectorizadas y calcula su similitud mediante un problema de transporte óptimo que determina el costo mínimo para ajustar las oraciones de un texto al otro [79].

- **Coherencia**

  Coherencia es una métrica para calificar la calidad de respuestas generados por Agentes

IA teniendo en cuenta la fluidez, consistencia y conexión de las ideas. Se enfoca en como se está organizado el contenido

Entre las métricas principales se encuentra:

- **Coherencia Estructural:** Mide la organización global del texto, el orden lógico y la transición entre ideas. Puede evaluarse con herramientas como UniEval [80].
- **Coherencia Interna:** Verifica que el contenido no presente contradicciones, inconsistencias temporales ni cambios abruptos de perspectiva. Se evalúa mediante UniEval bajo el criterio de Consistencia [80].
- **G-Eval:** Marco que mide la coherencia mediante criterios explícitos y pasos de razonamiento generados por un modelo de lenguaje. Los LLM actúan como jueces para estimar la continuidad lógica, claridad contextual y solidez argumentativa [81].

- **Framework RAGAS**

  RAGAS (Retrieval-Augmented Generation Assessment) es una biblioteca de código abierto para evaluar aplicaciones basadas en modelos de lenguaje. Se basa en comparar respuestas mediante llamadas recursivas a LLM, genera datos sintéticos, evalúa conversaciones de varios turnos y puede configurarse desde el código de la aplicación. Es compatible con métricas ampliamente usadas y con frameworks como LangChain, LlamaIndex o Haystack. Entre sus métricas se encuentran:

  - **Faithfulness (Fidelidad):** Mide la exactitud de la respuesta comprobando cuántas afirmaciones coinciden con el contexto. Se calcula dividiendo las afirmaciones correctas entre el total de afirmaciones de la respuesta, con un valor entre 0 y 1 [82].
  - **Answer Relevance (Relevancia de la Respuesta):** Evalúa qué tan bien la respuesta generada aborda la pregunta comparando la pregunta original con preguntas sintetizadas a partir de la propia respuesta mediante embeddings semánticos. A mayor similitud, mayor relevancia [82].
  - **Context Relevance (Relevancia del Contexto):** Identifica las oraciones esenciales del contexto para responder la pregunta y calcula la proporción entre estas oraciones

y el total de oraciones disponible [82].

2.9.3 Escalabilidad

Escalabilidad es la medida de que tan bien puede funcionar una IA para un crecimiento en usuarios, agentes y/o datos procesados. Es útil cuando se mantiene el número de tiempos de respuestas normales, una buena coordinación y buenos resultados.

- **Rendimiento con múltiples usuarios**

  - **Usuarios Concurrentes:** Número máximo de usuarios o agentes que el sistema puede manejar simultáneamente sin perder rendimiento. A medida que aumenta la cantidad de concurrencias, la latencia y la estabilidad pueden verse afectadas, aunque las arquitecturas cloud-native suelen mantener un comportamiento estable bajo alta demanda [83].
  - **Tiempo de respuesta bajo carga:** Mide la rapidez del sistema ante solicitudes concurrentes y refleja su capacidad para mantener un rendimiento ágil bajo alta demanda. Arquitecturas paralelas y asíncronas reducen este tiempo y lo mantienen estable incluso con múltiples solicitudes simultáneas [28].
  - **Throughput (Tasa de Procesamiento):** Cantidad de solicitudes procesadas por unidad de tiempo. Valores altos indican una gestión eficiente de usuarios simultáneos. En general, aumentar el número de agentes mejora el rendimiento, lo que es clave en entornos con alta concurrencia [28], [84].
  - **Utilización de Recursos:** Analiza el uso de CPU, memoria, red y almacenamiento en condiciones de alta carga. Las arquitecturas paralelas y asíncronas permiten aprovechar mejor estos recursos al optimizar la ejecución de tareas [28].

- **Rendimiento con Procesamiento de Documentos**

  El rendimiento en el procesamiento de documentos es un factor clave para la escalabili-

dad de los sistemas multiagente basados en LLM, especialmente al trabajar con grandes colecciones de contratos, reportes u otros documentos estructurados y no estructurados. A diferencia de la carga generada por usuarios concurrentes, el procesamiento documental introduce costos adicionales asociados a la lectura, segmentación, indexación y generación de respuestas a partir de múltiples fuentes.

En estos sistemas, la eficiencia del procesamiento documental depende principalmente de la capacidad para distribuir subtareas entre múltiples agentes y del costo computacional de operar sobre documentos de gran tamaño [28].

Estudios recientes indican que arquitecturas basadas en grafos de tareas o en delegación dinámica entre agentes reducen significativamente los tiempos de cómputo al paralelizar el análisis documental, lo que resulta esencial para mantener tiempos de respuesta aceptables a medida que aumenta el tamaño del corpus o el número de usuarios concurrentes [28].

Entre las métricas principales se encuentra:

- **Tiempo de procesamiento por documento:** Indica el tiempo total desde que se recibe el documento hasta que se genera la respuesta [51].
- **Throughput Documental:** Número de documentos analizados por unidad de tiempo bajo diferentes niveles de carga [51].
- **Escalado de Documentos (Document-Scaling Efficiency):** Evalúa si el sistema mantiene estabilidad del tiempo de procesamiento al aumentar la cantidad o tamaño de los documentos [51].
- **Utilización de recursos en tareas de análisis textual:** Monitorea el uso de CPU, memoria y almacenamiento durante operaciones intensivas de análisis y recuperación, lo que permite tener una visión más clara del consumo de recursos en cada etapa [51].

La escalabilidad de los sistemas depende de ellos para poder distribuir la carga entre los agentes. Esto es clave en procesamiento documental: El coste de búsqueda se incrementa conforme crece el corpus; particularmente importante para aproximaciones tipo RAG.

Por esta razón, se requieren arquitecturas distribuidas. La escalabilidad no solo se relaciona con el número de usuarios o agentes, sino también con la capacidad de procesar grandes volúmenes de información manteniendo un buen rendimiento.

CAPÍTULO III

MATERIALES Y MÉTODOS

El capítulo va a describir los materiales, las herramientas y los procedimientos metodológicos utilizados en la realización de esta investigación. Se desarrolla el tipo de abordaje y de estudio, así como las técnicas e instrumentos usados en la colecta, procesamiento y análisis de los datos, con el objetivo de responder a los objetivos propuestos.

3.1 Enfoque y Tipo de Investigación

Este estudio sigue un enfoque de investigación aplicada y se centra en el diseño e implementación de una arquitectura multiagente basada en el protocolo A2A para la extracción automatizada de cláusulas en contratos SaaS, utilizando modelos de lenguaje de código abierto.

La propuesta emerge de las dificultades señaladas en el capítulo anterior; ahí se puede ver que los usuarios tienen problemas interpretando un contrato extenso usando lenguaje jurídico complicado en estructuras repetitivas. En vez de limitarse al análisis teórico, la investigación presenta como solución el desarrollo tecnológico útil en facilitar entender, identificar, consultar a cláusulas relevantes.

3.1.1 Investigación aplicada en inteligencia artificial

La investigación aplicada en inteligencia artificial busca, en esencia, resolver problemas concretos mediante soluciones tecnológicas. En este caso, el proyecto propone un sistema que procesa contratos extensos y extrae automáticamente sus cláusulas más relevantes utilizando LLMs y una arquitectura multiagente, lo que, en la práctica, apunta a hacer este tipo de tareas

mucho más ágiles.

Como lo señala Zin et al.[14], los modelos de lenguaje a gran escala pueden comprender el lenguaje legal sin necesidad de entrenamiento previo específico y extraer simultáneamente tanto información puntual como cláusulas complejas en un proceso unificado, lo que los convierte en herramientas idóneas para este dominio.

Aplicar IA al análisis de contratos SaaS supone, sin embargo, un desafío técnico concreto: los documentos contractuales son extensos, poseen lenguaje jurídico técnico y presentan estructuras repetitivas que dificultan la recuperación precisa de información. En particular, la mayoría de los modelos de lenguaje rompen las correlaciones entre cláusulas al dividir los documentos en fragmentos de forma arbitraria, lo que impide comprender adecuadamente las relaciones entre secciones distantes del contrato [31].

- **Chunking semántico como estrategia de particionamiento**

  Para abordar este problema, la estrategia de chunking semántico resulta esencial. A diferencia del chunking de tamaño fijo que divide el texto en segmentos de longitud predeterminada o del chunking jerárquico que respeta únicamente la estructura formal del documento, el chunking semántico busca que cada fragmento represente una unidad de significado completa [50]. En documentos estructurados como contratos, donde las cláusulas contienen derechos, obligaciones y prohibiciones, preservar la coherencia semántica de cada fragmento resulta fundamental para garantizar una recuperación precisa.

  El agente almacenador en la implementación desarrollada hace segmentar el texto basándose en el parecido semántico de las oraciones sucesivas. Em vez de simplemente dividir el texto en segmentos con una cantidad dada de letras, se crean embeddings para las oraciones y se compara su similitud. Al disminuir esta similitud bajo el umbral definido (0.5 para esta implementación) este es considerado un cambio de tema y se empieza un nuevo fragmento (chunk).

  Un particionamiento inadecuado tiene consecuencias directas sobre la calidad de la recuperación: si una cláusula queda dividida artificialmente, el sistema no podrá reconstruir

su significado completo ante una consulta semántica posterior [50]. Por ello, el uso de segmentación basada en coherencia temática mejora la precisión en tareas de búsqueda jurídica asistida por IA.

La Figura 8 muestra el fragmento del agente almacenador que implementa el particionamiento semántico del texto extraído del PDF.

```python
# Generar embeddings para todas las oraciones en un solo paso (eficiente)
embeddings = model.encode(sentences, show_progress_bar=False)

# Calcular similitud coseno entre oraciones consecutivas
similarities = [
    util.cos_sim(embeddings[i], embeddings[i + 1]).item()
    for i in range(len(embeddings) - 1)
]

# Agrupar oraciones en chunks según el umbral de similitud
# Cuando la similitud cae por debajo del umbral → ruptura semántica → nuevo chunk
raw_chunks = []
current_chunk = [sentences[0]]

for i, score in enumerate(similarities):
    if score < similarity_threshold:
        raw_chunks.append(" ".join(current_chunk))
        current_chunk = []
    current_chunk.append(sentences[i + 1])
```

Figura 8.
Implementación del algoritmo de chunking semántico basado en similitud del coseno

- **Modelo de embeddings y representación vectorial**

  Como se observa en la Figura 8, los embeddings se generan por oración mediante model.encode() y la similitud coseno entre oraciones consecutivas se calcula con util.cos_sim(), determinando los puntos de ruptura semántica. Cada chunk resultante es posteriormente convertido en un vector numérico mediante el mismo modelo sentence-transformers/all-MiniLM-L6-v2, que produce vectores de dimensión 384.

  Este modelo fue seleccionado por su equilibrio entre rendimiento semántico y costo computacional. En este sentido, los embeddings capturan el significado contextual y las relaciones conceptuales del texto, lo que permite comparaciones semánticas genuinas. La similitud entre estos vectores se calcula mediante la distancia del coseno, una métrica ampliamente utilizada para este fin [3], [50].

3.1.2 Metodología de desarrollo basada en prototipado

Dado que el proyecto integra tecnologías emergentes como modelos de lenguaje, el protocolo A2A y bases de datos vectoriales, se adoptó una metodología iterativa basada en prototipado gestionada bajo el marco ágil Scrum. Scrum es una metodología ampliamente utilizada en el desarrollo de software que promueve la entrega incremental de valor mediante ciclos cortos de trabajo denominados sprints, fomentando la colaboración y la adaptación continua ante cambios [85].

Scrum se estructura en tres roles principales: el Scrum Master, encargado de facilitar el proceso y garantizar el cumplimiento de la metodología; el Product Owner, responsable de definir y priorizar los requerimientos; y el Scrum Team, equipo que ejecuta el desarrollo y cumple los objetivos de cada sprint [85].

Dado que es una investigación experimental, el protocolo A2A se desarrolló en iteraciones cortas de pruebas e integraciones. En medio de este proceso fue identificado un problema con respecto a RemoteA2aAgent, que tenía contaminación del estado interno debido a la reutilización de las sesiones. Esto afectaba también al LlMAgent. Ésto lo solucionamos con una sesión separada para cada mensaje.

Dado el contexto del proyecto, el Director de Tesis asumió el rol de Scrum Master, encargándose de la supervisión del proceso metodológico. Mientras tanto, el Investigador se desempeña como Product Owner (dueño del producto) y también como parte del equipo de desarrollo, a cargo de la definición de los requerimientos y la implementación del sistema.

- **Estructura de Sprints**

  El desarrollo se organizó en nueve sprints con entregas funcionales al final de cada uno:

Tabla II.

SPRINT 01 - INFRAESTRUCTURA BASE

Sprint 01 - Infraestructura Base		
Duración: 1 semana	**Estado:** Completado	
Historia de Usuario	**Actividad**	**Avance**
HU-01 Configuración del entorno	Crear entorno virtual .venv	100 %
	Instalar dependencias y frameworks	100 %
	Configurar .env (OpenRouter y Qdrant)	100 %
	Definir estructura del proyecto	100 %
HU-02 Conexión con Qdrant	Instanciar cliente Qdrant	100 %
	Crear colección (embeddings)	100 %
	Verificar conexión al iniciar	100 %
HU-03 Estructura A2A base	main.py del almacenador (puerto 8001)	100 %
	main.py del analizador (puerto 8002)	100 %
	AgentCard con skills definidas en cada agente, conforme a las especificaciones del protocolo A2A descritas en la sección 2.6	100 %

Tabla III.
SPRINT 02 - AGENTE ALMACENADOR

Sprint 02 - Agente Almacenador		
Duración: 2 semanas	**Estado:** Completado	
Historia de Usuario	**Actividad**	**Avance**
HU-04 Almacenamiento de PDF	Extracción de texto por página	100 %
	Generación de embeddings	100 %
	Almacenamiento en Qdrant con metadatos (filename, document_id, stored_at)	100 %
	Respuesta HTML o texto plano	100 %
HU-05 Consultas sobre documentos	Creación de función `list_documents()` en qdrant_storage.py	100 %
	Respuesta con tabla de documentos (HTML o texto)	100 %
HU-06 Almacenamiento de análisis	Almacenar análisis con referencia a document_id	100 %
	list_analyses() — listar análisis	100 %
	get_analysis() — recuperar análisis por ID	100 %
HU-07 Validación de duplicados	Verificar duplicados por hash (actualizar si existe)	100 %
	Responder con mensaje de actualización	100 %

Tabla IV.
SPRINT 03 - AGENTE ANALIZADOR

Sprint 03 - Agente Analizador		
Duración: 2 Semanas	**Estado:** Completado	
Historia de Usuario	**Actividad**	**Avance**
HU-08 Recuperación de chunks relevantes	QdrantRetriever con búsqueda por nombre y UUID	100 %
	Reconstrucción del texto desde chunks ordenados	100 %
	Manejo de documentos no encontrados o ambiguos	100 %
HU-09 Pipeline CrewAI de análisis	contract_analyzer_agent - análisis legal con criticidad ALTA/MEDIA/BAJA	100 %
	html_formatter_agent - salida HTML estructurada	100 %
	Referencias a cláusulas específicas	100 %
HU-10 Filtro semántico de chunks	Limitar chunks y seleccionar los más relevantes	100 %
	Log de chunks descartados	100 %

Tabla V.

SPRINT 04 - AGENTE ORQUESTADOR

Sprint 04 - Agente Orquestador		
Duración: 2 Semanas	**Estado:** Completado	
Historia de Usuario	**Actividad**	**Avance**
HU-11 Enrutamiento correcto de solicitudes	instruction con sección MECANISMO DE DELEGACIÓN	100 %
	Reglas de enrutamiento por palabras clave	100 %
	Mensaje de bienvenida con guía de uso	100 %
HU-12 Mensaje de bienvenida	Guía paso a paso en el primer mensaje	100 %
	Ejemplos de comandos aceptados	100 %
HU-13 Confirmación de flujo completo	Mensaje de resumen al completar la actividad	100 %
	Esperar a que el usuario solicite otra actividad a realizar	100 %

Tabla VI.
SPRINT 05 - INTERFAZ DE USUARIO

Sprint 05 - Interfaz de Usuario		
Duración: 1 Semana	**Estado:** Completado	
Historia de Usuario	**Actividad**	**Avance**
HU-14 Chat funcional con el orquestador	gr.ChatInterface con streaming palabra a palabra	100 %
	Sesión nueva por mensaje (evita estado contaminado de ADK)	100 %
	Historial visible en el frontend	100 %
HU-15 Subida de PDF	gr.File con restricción a .pdf	100 %
	PDF enviado como bytes al agente (no solo nombre)	100 %
	Confirmación visual al cargar el archivo	100 %
HU-16 Indicador de carga	Mensaje ”Processing...”mientras el runner itera eventos	100 %
	Tiempo estimado según tipo de operación	100 %

Tabla VII.
SPRINT 06 - INTEGRACIÓN Y ESTABILIZACIÓN

Sprint 06 - Integración y Estabilización		
Duración: 1 Semana	**Estado:** Completado	
Historia de Usuario	**Actividad**	**Avance**
HU-17 Flujo end-to-end completo	Subida y almacenamiento funcional	100 %
	Análisis recuperando de Qdrant	100 %
	Almacenamiento del análisis desde la interfaz	100 %
	Prueba con 5 contratos reales distintos	100 %
HU-18 Estabilidad del runner ADK	Sesión nueva por mensaje	100 %
	Iteración completa de eventos sin break prematuro	100 %
HU-19 Compatibilidad de versiones	Warnings de Pydantic silenciados	100 %
	Warnings experimentales de ADK silenciados	100 %
	Especificar versiones exactas en requirements.txt	100 %

Tabla VIII.

SPRINT 07 - SEGURIDAD Y CONFIGURACIÓN

Sprint 07 - Seguridad y Configuración		
Duración: 1 Semana	**Estado:** Completado	
Historia de Usuario	**Actividad**	**Avance**
HU-20 Variables de entorno seguras	Validar .env con python-dotenv en todos los agentes	100 %
	.gitignore con .env, _pycache__, *.pyc, session.db	100 %
	Documentar variables requeridas en README	100 %
HU-21 Límite de tamaño de PDF	Validación de tamaño antes de procesar	100 %
	Mensaje de error amigable al usuario	100 %

Tabla IX.
SPRINT 08 - TESTING Y CALIDAD

Sprint 08 - Testing y Calidad		
Duración: 1 Semana	**Estado:** Completado	
Historia de Usuario	**Actividad**	**Avance**
HU-22 Tests unitarios de herramientas	Tests para qdrant_storage.py (almacenar, recuperar, listar)	100 %
	Tests para qdrant_retriever.py (búsqueda por nombre y UUID)	100 %
	Tests para _extract_document_query() del analizador	100 %
HU-23 Tests de integración A2A	Test: orquestador → almacenador → respuesta HTML	100 %
	Test: orquestador → analizador → respuesta HTML	100 %
	Test: flujo completo con PDF real	100 %
HU-24 Casos borde	PDF sin texto (solo imágenes)	100 %
	PDF corrupto	100 %
	Qdrant sin conexión	100 %
	OpenRouter caído (error 502)	100 %
	Mensaje vacío del usuario	100 %

Tabla X.
SPRINT 09 - DOCUMENTACIÓN

Sprint 09 - Documentación		
Duración: 1 Semana	**Estado:** Completado	
Historia de Usuario	**Actividad**	**Avance**
HU-25 README completo	Arquitectura del sistema con diagrama	100 %
	Requisitos previos (Python, Qdrant, API keys)	100 %
	Pasos de instalación y ejecución	100 %

3.2 Diseño de la Arquitectura Multiagente

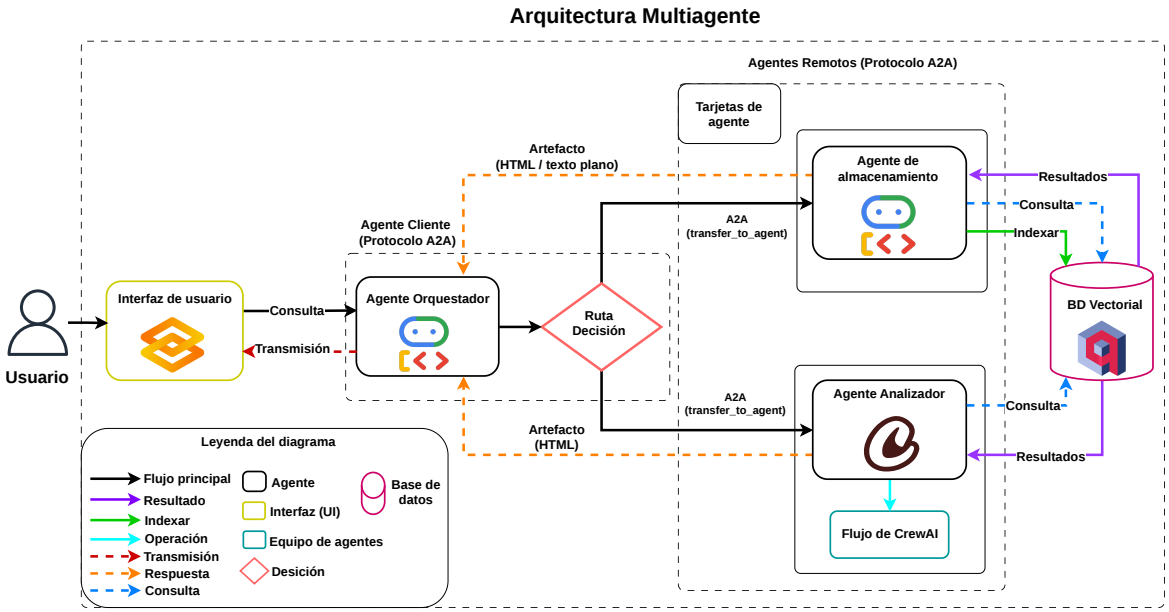


Figura 9.
Arquitectura multiagente

Como se observa en la Figura 9, el diseño de la arquitectura multiagente constituye el núcleo técnico del sistema. Su objetivo es distribuir las responsabilidades del proceso de extracción, almacenamiento, análisis y orquestación entre agentes especializados que colaboran mediante el protocolo A2A. La arquitectura implementada sigue un patrón jerárquico con un agente

coordinador central que delega tareas a agentes remotos especializados, correspondiente a la Arquitectura Jerárquica descrita en la sección 2.5.1.

3.2.1 Especificación de agentes

El sistema está compuesto por tres agentes, cada uno con un rol, framework y modelo LLM diferenciados. La Tabla XI resume su configuración:

Tabla XI.
ESPECIFICACIÓN DE LOS AGENTES DEL SISTEMA

Agente	Framework	Modelo LLM	Función principal
Orquestador	Google ADK	Gemini 2.5 Flash Lite	Recibe la consulta del usuario, la interpreta y delega al agente especializado correspondiente mediante transfer_to_agent.
Almacenador (8001)	Google ADK	Llama 4 Maverick	Procesa PDFs, hace chunks semánticos, crea embeddings y los almacena en Qdrant, para después poder consultar o listar.
Analizador (8002)	CrewAI	Grok 4.1 Fast	Por medio de sus 2 sub-agentes este recupera el texto de Qdrant, ejecuta el Crew de análisis legal y devuelve un reporte HTML con derechos, obligaciones y prohibiciones.

Nota: El modelo Llama 4 Maverick está configurado en el agente almacenador, pero en la versión actual no interviene en el procesamiento. Esta decisión de diseño responde a la necesidad de garantizar consistencia en las operaciones críticas de extracción, fragmentación y

almacenamiento, separando estas etapas de la variabilidad propia de la generación de texto por LLM, reservando al modelo un rol que podrá ampliarse en versiones futuras.

- **Agente Orquestador**

  El agente orquestador corresponde al Agente Coordinador descrito en la sección 2.5.2, cuya función consiste en gestionar y coordina las interacciones entre los distintos agentes del sistema [36]. Este diseño de orquestador centralizado que delega tareas en agentes especializados se alinea con los principios de estructuras de interacción centralizada descritos en la literatura [36], donde un agente central coordina y asigna subtareas basándose en información global.

  A la arquitectura propuesta, este agente lo implementamos en base a la plataforma Google ADK, utilizando como modelo el Gemini 2.5 Flash Lite con un mecanismo de fallback Llama 3.3 70b ambos accesibles mediante OpenRouter. Su funcionamiento se basa en reglas explícitas quien determina si el usuario quiere guardar, analizar, recobrar o catalogar los documentos, y en función de ello delega la solicitud al agente correcto usando transfer_to_agent sin procesar directamente los documentos.

  Un elemento importante en la configuración es que el orquestador no produce ni inventa herramientas de forma autónoma. En lugar de eso, se limita estrictamente al uso de transfer_to_agent, evitando recurrir a funciones no definidas como store_document, analyze_contract o fetch_document.

  Esta decisión, en la práctica, ayuda a disminuir la posibilidad de alucinación de herramientas, que, siendo sinceros, es un problema bastante común en sistemas basados en LLM cuando no existen controles estrictos.

- **Agente Almacenador**

  El Agente Almacenador amplía el rol del Agente Extractor descrito en la sección 2.5.2, incorporando las etapas de fragmentación semántica, vectorización y persistencia en una base de datos vectorial. Está construido sobre Google ADK con el modelo Llama 4 Maverick, accesible a través de OpenRouter, y expone sus capacidades mediante el protocolo A2A a través de una Agent Card publicada en el puerto 8001.

Este agente recibe los documentos de entrada en formato PDF desde el orquestrador. Primero extrae el texto página a página, y después fragmenta con chunking semántico. Durante éste proceso, se amontonan las frases que sigan a otra y cuya similitud coseno exceda un valor de umbral configurable (por defecto 0.5) haciendo así fragmentos temáticamente relacionados.

Los chunks resultantes son vectorizados utilizando el modelo all-MiniLM-L6-v2 de Sentence Transformers y luego almacenados en la base de datos vectorial Qdrant, desplegada localmente dentro de contenedores Docker.

Por otro lado, el agente responde a cinco tipos de solicitudes, que se identifican mediante palabras clave en la entrada del usuario: guardar un documento PDF, almacenar un análisis legal asociado, buscar un análisis por ID o nombre de archivo, consultar las estadísticas del sistema y listar los documentos que cuentan con análisis registrados.

Al nivel de la implementación, no se definen herramientas explícitas. La ejecución de toda la lógica ocurre directo en el executorque produce las respuestas como output en formato html hacia el orquestador. Aunque es configurado con el modelo LLama 4 Maverick el agente ,en la version actual el LLM no participa en el procesando: La logica de extracción/segmentación/almacenar esta hecha totalmente en codigo.

Esto asegura comportamientos bastante más deterministas en los procesos críticos y, además, deja la puerta abierta para incorporar el modelo en versiones posteriores, ya sea para tareas como la validación semántica o la clasificación de contratos.

Y por último, el sistema presenta un mecanismo anti-duplicado basado en hash SHA-256. Antes de almacenar el documento se calcula la huella digital del mismo y se le hace una consulta a Qdrant. Si es afirmativo, reutilizar el ID existente; borrar todos los chunks anteriores y reemplazar con los nuevos, manteniendo la trazabilidad sin duplicar información. Por otro lado, hay una segunda colección que almacena y recupera el análisis legal asociado con cada documento usando su identificador.

- **Agente Analizador**

  El agente analizador corresponde al rol de Agente Analizador descrito en la sección 2.5.2,

y constituye el centro de inteligencia del sistema. A diferencia del agente almacenador [39], está construido sobre CrewAI en lugar de Google ADK, y utiliza los modelos Grok 4.1 Fast, Gemini 2.5 Flash Lite y Qwen3 VL 30B A3B Thinking vía OpenRouter. Expone sus capacidades mediante el protocolo A2A a través de una Agent Card publicada en el puerto 8002.

Internamente, la arquitectura está estructurada en cuanto a un equipo de dos subagentes que tiene roles diferenciados: Un analista legal de contratos y un estructurador de reportes en HTML. Esta distribución interna de roles es un ejemplo de la Arquitectura de Equipo descrita en la sección 2.5.1, donde los agentes especializados cooperan para producir resultados que superan las capacidades de un único agente [10], [37].

A diferencia del agente almacenador, el LLM participa de forma activa en los tres subagentes a lo largo de todo el proceso de análisis. El flujo de trabajo sigue una secuencia bastante clara: primero, el Analista identifica derechos, obligaciones y prohibiciones, relacionándolos con las cláusulas relevantes y su nivel de criticidad (ALTA, MEDIA o BAJA).

Dicho eso, el Formateador toma esa salida y la transforma en un reporte HTML ya estructurado, listo para su presentación.

El agente usa QdrantRetriever, un módulo de solamente lectura para recuperar el texto del contrato. Lectura que se conecta a la misma instancia Qdrant del agente almacenador sin escribir/modificar nada. Este módulo detectará automáticamente si la consulta recibida es un UUID o un nombre de fichero, y realiza el correspondiente proceso de búsqueda. Además, también maneja los casos de ambigüedad; por ejemplo, cuando existen varios documentos con el mismo nombre, el sistema notifica al usuario y le solicita el UUID exacto del documento para poder realizar la verificación de forma correcta.

El identificador de documento puede especificarse en 4 maneras distintas: el UUID completo, el nombre del archivo con la terminación ”.pdf”, con el nombre en comillas y/o detrás de las palabras clave analisa o revisa. Los chunks recuperados antes del analysis pasan por un filtro para eliminar: fragmentos vacios, muy pequeños (dos palabras o me-

nos), sin carácteres alfabéticos o dominado por las mayusculas (mas del 70 %) suelen ser titulos o ruido. Solo los fragmentos validos son enviados al Crew de Análisis.

Una vez completado el análisis, el agente incluye en la respuesta una instrucción lista para copiar que permite al usuario almacenar el resultado en el agente almacenador, vinculándolo al documento original mediante su ID.

3.2.2 Protocolo A2A para coordinación

El protocolo A2A es el canal de comunicación que conecta al orquestador con los agentes especializados. En la implementación desarrollada, cada agente especializado expone un servidor A2AStarletteApplication, mediante el cual recibe solicitudes enviadas por el agente orquestador. Cada agente declara sus capacidades a través de su AgentCard, donde se especifican sus habilidades, formatos de entrada y salida y soporte para streaming siguiendo la estructura descrita en la sección 2.6.

Cada AgentCard define las capacidades del agente: nombre, versión, habilidades, entradas, salidas y soporte para streaming [40]. El agente almacenador acepta application/pdf y text/plain como entrada, mientras que el agente analizador genera respuestas en text/html, listas para visualización en HTML.

- **Flujo de coordinación entre agentes durante una tarea de análisis**

  El siguiente flujo describe, paso a paso, la secuencia de interacciones cuando un usuario carga un PDF y solicita su análisis:

  a) El usuario empieza adjuntando un documento PDF en Gradio y escribe: Almacena el documento con el nombre (nombre).

  b) El orquestador recibe el mensaje junto con el PDF en formato binario, codificado como blob multimodal de Google ADK.

  c) El orquestador detecta la intención de almacenar el documento y delega la tarea al agente almacenador mediante la operación transfer_to_agent del protocolo A2A.

d) El agente almacenador extraerá el texto de un documento y creará los chunks semanticos, hace embeddings y los almacena en Qdrant. Como resultado, devuelve un reporte en HTML con el ID generado.

e) El usuario solicita el análisis del documento indicando su nombre en formato .pdf.

f) El orquestador identifica la intención de análisis y delega la tarea al agente analizador.

g) El agente analizador recupera los chunks desde Qdrant usando QdrantRetriever, aplica un filtrado, reconstruye el texto completo y, posteriormente, ejecuta el Crew de CrewAI, coordinando así las tareas de análisis de forma ordenada.

h) El Crew procesa la solicitud y elabora un reporte en HTML con derechos, obligaciones y prohibiciones detectadas. Por último el orquestrador pasa a presentar al usuario el resultado usando Gradio.

- **Manejo de sesiones y estado**

  Una decisión técnica bastante relevante fue la gestión de sesiones dentro del orquestador. En este caso, se implementó una estrategia de sesión nueva para cada mensaje. La razón es que, al utilizar RemoteA2aAgent, una vez finalizada una delegación, la sesión queda con un estado interno **contaminado** y, si se reutiliza, el agente orquestador deja de generar una nueva respuesta en el siguiente turno.

  Para evitar este comportamiento, se optó por crear una sesión nueva en cada interacción y, al mismo tiempo, delegar el historial visible de la conversación al componente ChatInterface de Gradio.

3.3 Desarrollo de la Aplicación Web

El frontend de una aplicación web está conectado al backend (agentes, protocolo A2A y base de datos vectorial). Una interfaz simple permite que los usuarios carguen contratos SaaS para recibir automáticamente un análisis estructural de las cláusulas relevantes.

La interfaz fue construida con Gradio, una biblioteca de código abierto en Python que permite crear interfaces interactivas para modelos de machine learning, aplicaciones de inteligencia artificial o cualquier función programada. Esta herramienta facilita el desarrollo de interfaces mediante pocas líneas de código, sin requerir conocimientos avanzados en HTML, CSS o JavaScript [86].

Por su parte, el backend opera mediante dos servidores Starlette independientes expuestos como agentes A2A. El agente orquestador se ejecuta como cliente A2A y coordina las interacciones entre ellos.

3.3.1 Requisitos funcionales y no funcionales

Los requisitos del sistema se clasificaron en funcionales, referidos a lo que el sistema debe hacer, y no funcionales, referidos a cómo debe comportarse. Ambas categorías fueron validadas durante el desarrollo iterativo del sistema y se detallan en la Tabla XII y Tabla XIII.

Tabla XII.
REQUISITOS FUNCIONALES DEL SISTEMA

ID	Requisito	Descripción
RF01	Agregar archivo PDF	El sistema debe permitir la carga de archivos PDF de hasta 20 MB desde la interfaz web.
RF02	Base de datos de vectores	El sistema debe dividir el PDF en chunks semánticos, generar embeddings con all-MiniLM-L6-v2 y almacenarlos en Qdrant con un ID único.
RF03	Limpieza de duplicados	El sistema debe detectar duplicados mediante hash SHA-256 y, en lugar de crear nuevos registros, actualizar los chunks existentes cuando corresponda.

La tabla continúa en la siguiente página

Continuación de la Tabla XII

ID	Requisito	Descripción
RF04	Revisión de cláusulas	El sistema debe analizar contratos almacenados para identificar derechos, obligaciones y prohibiciones, generando un reporte HTML estructurado que incluya el nivel de criticidad y la cláusula asociada.
RF05	Visualización de análisis	El sistema debe permitir la recuperación y visualización de análisis previamente generados usando el nombre del archivo o el UUID del documento.
RF06	Listado de documentos	El sistema debe listar los documentos almacenados e indicar cuáles poseen análisis registrados.
RF07	Chat en lenguaje natural	El sistema debe responder a instrucciones del usuario escritas en lenguaje natural a través de la interfaz de chat, sin usar comandos técnicos.
RF08	Arquitectura multiagente	El sistema debe operar mediante tres agentes especializados (orquestador, almacenador y analizador) que se comunican entre sí exclusivamente mediante el protocolo A2A.

Tabla XIII.
REQUISITOS NO FUNCIONALES DEL SISTEMA

ID	Requisito	Descripción
RNF01	Disponibilidad	Los servidores del agente almacenador y analizador e interfaz gráfica deben estar activos de forma simultánea durante el uso del sistema.

La tabla continúa en la siguiente página

Continuación de la Tabla XIII

ID	Requisito	Descripción
RNF02	Seguridad	Las API keys deben gestionarse mediante variables de entorno (.env) y nunca deben estar expuestas en el código fuente.
RNF03	Trazabilidad	El sistema debe registrar automáticamente el tiempo de respuesta de cada operación en metrics.csv para su posterior evaluación.
RNF04	Facilidad de uso	La interfaz debe poder utilizarse con instrucciones en lenguaje natural, sin necesidad de conocimientos técnicos.
RNF05	Movilidad del sistema	Qdrant debe ejecutarse en Docker para asegurar independencia del sistema operativo y del entorno de despliegue.
RNF06	Rendimiento	El tiempo de respuesta para operaciones de almacenamiento no debe superar los 10 segundos. El tiempo de análisis queda excluido de este límite dado el procesamiento intensivo de CrewAI.

3.3.2 Integración de LLMs y base de datos

El componente principal en la frontera entre los LLMs y las bases de datos representa uno de los componentes centrales del sistema desarrollado. Esta integración hace posible al agente acceder a informaciones estructuradas; generar respuestas contextualmente pertinentes y así combinar las habilidades de razonamiento de los LLMs junto con la precisión de los datos archivados.

Para manejar el acceso a diversos modelos de lenguaje se usó OpenRouter como una capa de enrutamiento para lograr la mejor selección y configurar el modelo más apropiado.

- **Selección de modelos LLM mediante OpenRouter**

  OpenRouter es una plataforma que actúa como agregador e intermediario entre diferentes modelos de inteligencia artificial. Su función es ofrecer a los usuarios un acceso centralizado a múltiples modelos de lenguaje, simplificando la integración y el manejo de diversos proveedores de IA [87].

  En esta arquitectura, los modelos de lenguaje se utilizan a través de la API de OpenRouter, que funciona como una puerta de entrada unificada tanto para modelos open source como comerciales. Esta decisión permite desacoplar el sistema de un único proveedor y, además, facilita la sustitución de modelos sin necesidad de modificar la arquitectura general del sistema.

  La selección de los modelos para cada agente responde a criterios funcionales específicos. El agente orquestador requiere baja latencia para el enrutamiento de solicitudes, por lo que utiliza Gemini 2.5 Flash Lite, con Llama 3.3 70B como mecanismo de respaldo. El agente almacenador emplea Llama 4 Maverick como modelo base de su arquitectura, sin embargo, en la versión actual la lógica de procesamiento se ejecuta directamente en el executor mediante código estructurado, reservando al LLM un rol que podrá ampliarse en versiones futuras.

  Finalmente, el agente analizador demanda mayor capacidad de razonamiento para el análisis jurídico de los contratos, por lo que utiliza Grok 4.1 Fast, Gemini 2.5 Flash Lite y Qwen3 VL 30B A3B Thinking, que son modelos razonadores.

  Se accede a los modelos mediante LiteLLM, una capa que unifica las llamadas a distintos proveedores que, además, utilizan una interfaz compatible con la API de OpenAI. Esto, en la práctica, facilita bastante la integración de varios LLMs y permite cambiar de modelo en el futuro sin tener que modificar la lógica del sistema.

- **Configuración de Qdrant y colecciones vectoriales**

Qdrant es una base de datos vectorial de código abierto especializada en el almacenamiento y la búsqueda de vectores de alta dimensión para realizar búsquedas de similitud de forma eficiente. Puede desplegarse tanto en entornos locales como en la nube y está diseñada para manejar grandes volúmenes de datos y consultas complejas. Además, admite distintos tipos de representaciones vectoriales utilizadas en sistemas modernos de redes neuronales [59].

En la arquitectura propuesta, Qdrant se despliega localmente mediante un contenedor Docker expuesto en el puerto 6333. El sistema utiliza esta base de datos para almacenar vectores y sus metadatos asociados en formato JSON. Durante la inicialización del componente QdrantStorageManager, se crean automáticamente dos colecciones:

- **Colección de documentos:** Almacena los chunks del texto contractual con sus embeddings de dimensión 384 y metadatos.
- **Colección de análisis:** Almacena los reportes HTML generados por el agente analizador y los vincula al ID del contrato correspondiente, de modo que cada análisis quede correctamente asociado a su documento.

Pero este mecanismo permite la recuperar de cláusulas relacionadas con el mismo concepto, todavía cuando están expresadas en sinónimos o con formulación diferente, sobreponiéndose a la limitación del único aspecto de búsqueda basada en palabras clave.

- **Interfaz de usuario con Gradio y registro de métricas**

  La interfaz se implementó utilizando ChatInterface de Gradio, que ofrece un entorno conversacional bastante práctico, con historial de conversaciones, ejemplos predefinidos y también soporte para la carga de archivos.

  Por otro lado, el componente gr.File limita la carga de archivos PDF hasta 20 MB. De igual forma, el feedback del agente se genera de forma incremental mediante streaming, lo que permite ver una especie de representación visual del razonamiento mientras se va construyendo y, al mismo tiempo, mejora la percepción de velocidad para el usuario.

  El sistema implementa además un mecanismo automático de registro de métricas para monitorear el desempeño de los agentes durante la ejecución de las operaciones. Cada ope-

ración completada genera una línea en el archivo metrics.csv con los campos: fecha/hora, agente, operación, documento, tiempo y estado. Este archivo constituye la fuente de datos primaria para la evaluación de eficiencia descrita en la sección 3.4.2.

3.4 Métricas de Evaluación

La evaluación del sistema se estructura en tres dimensiones complementarias que responden directamente a los criterios de calidad definidos en el marco teórico (sección 2.9): la precisión semántica de los análisis generados, la eficiencia temporal del sistema y un análisis comparativo que considera dos variables independientes: el modelo LLM utilizado y las prestaciones del hardware del equipo de ejecución. Adicionalmente, los resultados de la evaluación semántica automatizada serán contrastados con una evaluación humana, con el fin de validar la confiabilidad de las puntuaciones obtenidas.

Si bien la sección 2.9 expone un conjunto amplio de métricas disponibles para la evaluación de agentes de IA, la presente investigación selecciona aquellas que se ajustan a las condiciones del sistema desarrollado, el cual procesa un documento a la vez y opera sobre un corpus de 10 contratos SaaS reales de diferentes plataformas, conforme al alcance definido en la sección 1.3.

Para la evaluación de la precisión semántica se utiliza el framework DeepEval, que implementa G-Eval como mecanismo de puntuación subyacente mediante un LLM actuando como juez, permitiendo calcular las métricas seleccionadas sin requerir respuestas de referencia elaboradas manualmente [81]. Para la eficiencia se registra automáticamente un log de métricas operacionales por agente. Ambos instrumentos se aplican de forma sistemática sobre los 10 contratos de prueba en cada condición comparativa.

3.4.1 Precisión en identificación de cláusulas

La precisión semántica evalúa la calidad de los análisis generados por el sistema en términos de fidelidad al documento original, relevancia respecto a la consulta del usuario y exactitud en

la recuperación de los fragmentos contractuales pertinentes. La evaluación se realiza mediante DeepEval, framework que implementa G-Eval como juez LLM para puntuar cada métrica de forma automatizada sobre los análisis generados para cada uno de los 10 contratos de prueba. La Tabla XIV describe las cuatro métricas utilizadas:

Tabla XIV.
MÉTRICAS DE PRECISIÓN SEMÁNTICA PARA EVALUACIÓN DEL ANÁLISIS

Métrica	**Qué evalúa**	**Aplicación en el sistema**
Fidelidad	Proporción de afirmaciones de los análisis verificables en el contrato original.	Se valida que el agente no invente cláusulas que no están presentes en el documento.
Relevancia	Mide qué tan bien la respuesta trata la consulta del usuario.	Verifica que el análisis excluya cualquier clausula no demandada.
Precisión de Contexto	Proporción de fragmentos recuperados que son realmente relevantes para la consulta.	Evaluar la precisión del agente al extraer los distintos segmentos a partir de un extracto de contrato, lo que permite tener una idea más clara de qué tan bien está funcionando en la práctica.
Recall de Contexto	Evalúa qué tan bien el sistema logra recuperar la información contractual importante en su totalidad.	En este caso, hay que tener cuidado con no omitir cláusulas relevantes durante el proceso de extracción de información.

La métrica Fidelidad verifica que las afirmaciones presentes en el análisis generado sean verificables en el texto original del contrato, asegurando que el agente analizador no introduzca información inexistente [82]. Por su parte, Relevancia mide en qué medida la respuesta generada

aborda directamente la consulta del usuario, evitando desviaciones hacia cláusulas no solicitadas [82].

La Precisión de Contexto evalúa la proporción de fragmentos recuperados por el agente extractor que son genuinamente relevantes para la consulta, mientras que Recall de Contexto cuantifica qué porción de los fragmentos relevantes disponibles en el contrato fueron efectivamente recuperados, garantizando que no se omitan cláusulas importantes [82]. En conjunto, estas cuatro métricas permiten una evaluación integral tanto de la calidad generativa del agente analizador como de la precisión recuperativa del agente extractor.

Con el objetivo de validar la confiabilidad de las puntuaciones obtenidas mediante G-Eval, los resultados de cada métrica serán contrastados con una evaluación humana realizada sobre el mismo conjunto de contratos de prueba. Esta comparación permite determinar el grado de correlación entre el juicio automatizado del LLM y el criterio humano, siguiendo el enfoque de validación propuesto por Liu et al. [81], quienes fundamentan G-Eval precisamente en su alta alineación con evaluaciones humanas.

De este modo, la evaluación humana actúa como referencia de validación externa, fortaleciendo la credibilidad de los resultados automatizados obtenidos y aportando evidencia sobre la fiabilidad del sistema de evaluación adoptado.

3.4.2 Eficiencia en consultas y escalabilidad

La eficiencia del sistema se evalúa mediante el registro automático de métricas operacionales generadas durante el procesamiento de cada contrato. La función de registro almacena en el archivo metrics.csv los siguientes campos por cada operación ejecutada: timestamp, agente, operacion, documento, tiempo_s y status.

Este registro permite calcular estadísticos descriptivos para cada tipo de operación y para cada agente de forma individual, ofreciendo una visión granular del comportamiento temporal del sistema. La Tabla XV resume las métricas de eficiencia consideradas:

Tabla XV.
MÉTRICAS DE EFICIENCIA OPERACIONAL DEL SISTEMA

Métrica	**Dimensión**	**Instrumento de medición**
Latencia por operación (s)	Tiempo de respuesta	Registro automático en metrics.csv con los campos: timestamp, agente, operacion, documento, tiempo_s, status.
Tiempo por fase	Eficiencia por agente	A continuación se presenta el desglose de ambos parámetros: almacenamiento, análisis y orquestación.
Estadísticos descriptivos	Variabilidad temporal	Se calculan la media, mediana, mínimo, máximo y desviación estándar sobre los 10 contratos de prueba.

El análisis de la latencia por operación permite identificar posibles cuellos de botella dentro del flujo de trabajo. En un entorno multiagente, esto ayuda a ver con más claridad dónde se concentra el tiempo de procesamiento: si en la fase de diseño del almacenamiento y vectorización (Agente 2), o más bien en la etapa de análisis y generación de respuestas (Agente 3), que, dependiendo del caso, puede ser la más exigente.

Los estadísticos descriptivos obtenidos a partir de los 10 contratos de prueba permiten evaluar la estabilidad del sistema y detectar variaciones asociadas a diferencias en la extensión o complejidad de los documentos procesados.

3.4.3 Análisis comparativo

El análisis comparativo complementa las dimensiones anteriores con el objetivo de determinar el impacto de dos variables independientes sobre el desempeño del sistema: el modelo

LLM utilizado para el análisis y el entorno de hardware donde se ejecuta la aplicación. Se evalúan tres modelos LLM: Grok 4.1 Fast (xAI) y Gemini 2.5 Flash Lite (Google), ambos modelos propietarios y Qwen3 VL 30B A3B Thinking (Alibaba), modelo open-source, bajo una licencia Apache 2.0

La selección de estos tres modelos se fundamenta en su posicionamiento en el ranking de OpenRouter en la categoría Legal [87], lo que garantiza que la comparativa se realiza sobre modelos con desempeño verificado en tareas de procesamiento jurídico.

Los equipos fueron seleccionados para representar dos niveles distintos de capacidad computacional. El Equipo A corresponde a una configuración de gama media-baja, con recursos más limitados en memoria RAM y gráficos integrados de menor rendimiento. Mientras que, el Equipo B representa una configuración de gama media, con mayor capacidad de memoria y gráficos integrados más potentes, lo que le da un mejor desempeño general.

Esto permite evaluar si las diferencias en el entorno de ejecución impactan de forma significativa en los tiempos de procesamiento del sistema. Las características de los equipos de hardware utilizados se detallan en la Tabla XVI:

Tabla XVI.
CARACTERÍSTICAS DE LOS EQUIPOS DE HARDWARE UTILIZADOS EN LA EVALUACIÓN

Equipo	Marca	Procesador	Tarjeta gráfica	RAM	Almacenamiento
A	Lenovo	Intel Core i5 (10th Gen)	128 MB (UHD Graphics)	8 GB	477 GB SSD
B	Acer	Intel Core i5 (11th Gen)	128 MB (Iris Xe Graphics)	12 GB	238 GB SSD

Los tres modelos LLM van cada uno sobre los dos entornos de hardware, procesando los diez contratos de prueba con cada combinación hacen un total de sesenta análisis. En cada ejecución se registran simultáneamente las métricas de precisión semántica mediante G-Eval y las métricas de eficiencia operacional mediante metrics.csv, permitiendo comparar tanto la calidad del

análisis como el rendimiento temporal bajo cada condición. La Tabla XVII resume las variables y criterios de comparación:

Tabla XVII.
VARIABLES DEL ANÁLISIS COMPARATIVO

Variable	**Valores evaluados**	**Criterio de comparación**
Modelo LLM	Grok 4.1 Fast; Gemini 2.5 Flash Lite; Qwen3 VL 30B A3B Thinking	Puntuaciones de Fidelidad, Relevancia, Precisión de Contexto y Recall de Contexto (DeepEval) y ejecutados en ambos equipos sobre los 10 contratos.
Entorno de hardware utilizado	Equipo A y Equipo B	Se registraron la latencia por operación y métricas de calidad semántica en cada equipo para los tres modelos LLM, lo que dio como resultado un total de 60 análisis.

Esta estructura cruzada permite responder de forma independiente qué modelo LLM genera análisis de mayor calidad en el contexto específico de extracción de cláusulas en contratos SaaS, y si las diferencias en capacidad de cómputo entre el Equipo A y el Equipo B impactan significativamente en los tiempos de procesamiento de los tres modelos. Los resultados de ambas comparativas se presentan de forma conjunta en el Capítulo 4 para facilitar su interpretación cruzada.

CAPÍTULO IV

RESULTADOS Y ANÁLISIS

El presente trabajo muestra los resultados obtenidos tras la implementación y evaluación del sistema multiagente diseñado para la extracción automatizada de cláusulas en contratos SaaS. A lo largo de las siguientes secciones se describen las especificaciones del sistema tal como fue desplegado, el comportamiento que tuvo durante las pruebas y un análisis de su desempeño general, considerando tanto los aciertos como los aspectos que podrían seguir mejorando.

4.1 Especificaciones del Sistema

El presente capítulo expone los resultados obtenidos tras la implementación y evaluación del sistema multiagente desarrollado para la extracción automatizada de cláusulas en contratos SaaS. La descripción que sigue corresponde al sistema tal como fue desplegado durante las pruebas, incluyendo su arquitectura final, el flujo de procesamiento adoptado y las decisiones técnicas que determinaron su comportamiento durante la evaluación.

4.1.1 Diagrama de la arquitectura multiagente

El sistema implementado sigue una arquitectura jerárquica compuesta por tres agentes especializados que se comunican exclusivamente mediante el protocolo A2A. En la cima de esta jerarquía se sitúa el agente orquestador, construido sobre Google ADK con el modelo Gemini 2.5 Flash Lite, cuya única responsabilidad es interpretar la intención del usuario y delegar la solicitud al agente especializado correspondiente mediante el mecanismo transfer_to_agent. Este agente no procesa documentos ni genera análisis de forma directa; su rol es exclusivamente coordinador.

El agente de almacenamiento, llamado también almacenador, es construido sobre Google ADK y expuesto a través del puerto 8001. Toma documentos PDF del orquestador. Extrae el texto de estos documentos. Fragmenta ese texto utilizando el chunking semántico. Crear embeddings a partir de estos fragmentos usando el modelo all-MiniLM-L6-v2 y almacenarlos en la base de datos Qdrant vectorial.

El agente analizador, expuesto en el puerto 8002 y construido sobre CrewAI, recupera los chunks almacenados mediante QdrantRetriever y ejecuta un pipeline de análisis legal que produce un reporte HTML estructurado con derechos, obligaciones y prohibiciones identificados en el contrato. La Figura 10 presenta el diagrama completo de la arquitectura desplegada.

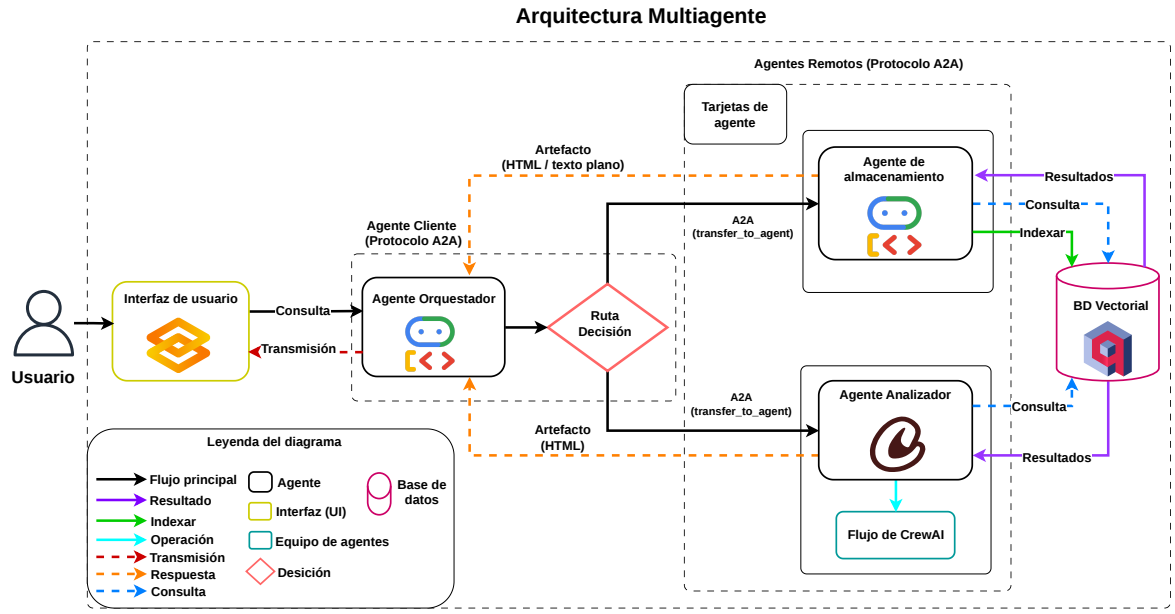


Figura 10.
Arquitectura del sistema multiagente implementado

Una decisión técnica relevante que influyó directamente en el comportamiento del sistema durante las pruebas fue la gestión de sesiones en el orquestador. Se implementó la estrategia de crear una sesión nueva por cada mensaje, dado que la reutilización de sesiones con RemoteA2aAgent generaba contaminación del estado interno, impidiendo que el orquestador generara respuestas en turnos subsecuentes. Esta solución delegó el historial visible de la conversación al componente ChatInterface de Gradio, manteniendo la experiencia del usuario sin sacrificar la

estabilidad del sistema.

4.1.2 Flujo de procesamiento de contratos

El procesamiento de un contrato SaaS en el sistema sigue una secuencia de dos etapas diferenciadas: almacenamiento y análisis. Ambas se inician desde la interfaz de chat y son coordinadas por el agente orquestador, que determina el destino de cada solicitud a partir del análisis de palabras clave en el mensaje del usuario.

El usuario carga un PDF a través de la fase de almacenamiento y, posteriormente, lo guarda mediante la interfaz. En ese momento, el orquestador recibe el archivo como un blob multimodal, el cual es delegado al agente almacenador mediante A2A.

A partir de ahí, el agente primero extrae el texto del documento, luego genera embeddings por frase y, finalmente, detecta posibles cambios en el tópico, lo que permite organizar mejor la información procesada.

Los chunks resultantes son almacenados en Qdrant junto con metadatos que incluyen el nombre del archivo, un identificador único de documento y la marca de tiempo del almacenamiento.

Antes de almacenar, el sistema genera un hash SHA-256 para comprobar si ya existe un documento igual y, con eso, evitar duplicados. Si el contrato ya está registrado, los chunks se actualizan en lugar de crear una nueva entrada, lo cual mantiene la base de datos más limpia.

Luego, cuando el usuario solicita un análisis de un contrato mediante su nombre o UUID, el orquestador delega la tarea al agente analizador. Este recupera los chunks desde Qdrant, filtra el contenido de baja calidad y, finalmente, envía el texto al pipeline de CrewAI para continuar con el procesamiento.

Este pipeline ejecuta de forma secuencial dos sub-agentes: un analista legal que analiza y clasifica los hallazgos por nivel de criticidad ALTA, MEDIA o BAJA, y un formateador que produce el reporte HTML final. El sistema registra automáticamente el tiempo de cada opera-

ción en el archivo metrics.csv, constituyendo la fuente de datos primaria para la evaluación de eficiencia presentada en la sección 4.4.1.

4.2 Implementación

Documentamos las decisiones de implementación que influyeron directamente a esta sección. El desempeño y la calidad de los resultados obtenidos, haciendo hincapié en la selección de modelo de un nuevo lenguaje y el diseño de la intefaz de usuario.

4.2.1 Entrenamiento y ajuste de LLMs para lenguaje jurídico

El sistema no incorpora ningún proceso de fine-tuning, reentrenamiento ni ajuste adicional sobre los modelos de lenguaje utilizados. Esta decisión no responde a una limitación técnica, sino a una elección metodológica fundamentada en evidencia reciente: Zin et al. [14], demuestran que los modelos de lenguaje a gran escala comprenden el lenguaje legal sin necesidad de entrenamiento previo específico, gracias al conocimiento adquirido durante su preentrenamiento en textos generales, y que son capaces de extraer simultáneamente tanto información puntual como cláusulas complejas en un proceso unificado.

En este sentido, someter los modelos a un proceso de ajuste adicional no solo habría incrementado el costo computacional del proyecto, sino que podría haber restringido innecesariamente su capacidad de generalización sobre contratos de distintas plataformas y estructuras.

La selección del modelo se basó en el ranking de la categoría “Legal” de OpenRouter: Grok 4.1 Fast, Gemini 2.5 Flash Lite y Qwen3 VL 30B A3B Thinking. En general, son modelos adecuados para tareas del ámbito legal.

Los dos primeros son propietarios, mientras que el tercero es open source bajo la licencia Apache 2.0. Todos se integran a través de LiteLLM en OpenRouter, lo que unifica su uso y, además, permite cambiar entre ellos sin afectar la lógica del sistema.

4.2.2 Interfaz web para consultas de usuarios

La interfaz de usuario fue construida con ChatInterface de Gradio, una biblioteca de código abierto en Python que permite crear interfaces interactivas sin requerir conocimientos avanzados de desarrollo web. La elección de Gradio responde directamente al requisito no funcional RNF04 definido en la sección 3.3.1: la interfaz debe poder operarse con instrucciones en lenguaje natural sin requerir formación técnica por parte del usuario.

Los tres elementos de la interfaz son un chat con historial visible a los lados, un cargador de archivos PDF limitados a 20 MB y un indicador mostrando ”Processing...”mientras el runner ADK está activo.

Las respuestas del agente se emiten mediante streaming palabra a palabra, lo que permite al usuario visualizar el análisis mientras se genera, reduciendo la percepción de espera en operaciones que pueden extenderse varios segundos.

Cuando el usuario inicia sesión, el orquestador muestra un mensaje de bienvenida y, además, ofrece una guía paso a paso con ejemplos de uso, lo que evita la necesidad de documentación externa.

Por otro lado, la Figura 11 muestra la interfaz durante una sesión de análisis.

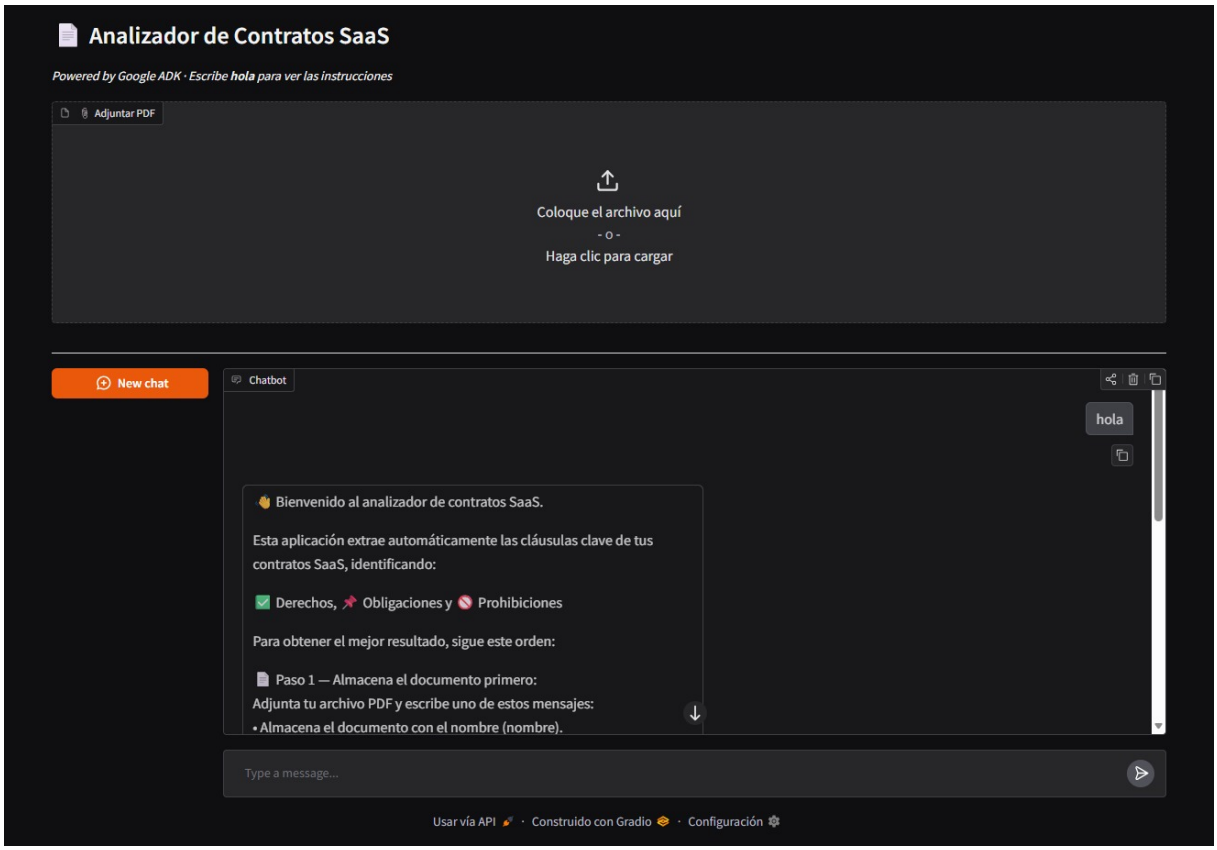


Figura 11.
Interfaz de usuario diseñada con gradio

4.3 Pruebas y Validación

Las pruebas se realizaron sobre un corpus de 10 contratos SaaS reales pertenecientes a diferentes plataformas, conforme al alcance definido en la sección 1.3. Cada contrato fue procesado bajo las seis condiciones del análisis comparativo resultantes de combinar los tres modelos LLM con los dos entornos de hardware descritos en la Tabla XVI del Capítulo 3, generando un total de 60 análisis evaluados mediante G-Eval y validados con evaluación humana.

4.3.1 Casos de uso con contratos SaaS reales

La Tabla XVIII presenta las características del corpus de prueba utilizado en la evaluación. Para cada contrato, se indica el número de páginas, el tamaño del archivo en KB, el total de chunks generados tras el procesamiento semántico y el número de chunks que superaron el

filtro de ruido estructural aplicado por el agente analizador antes de proceder con el análisis.

Tabla XVIII.
CARACTERÍSTICAS DEL CORPUS DE CONTRATOS SAAS

Nombre	**Descripción**	**N° páginas**	**Tamaño (KB)**	**Total chunks**	**Chunks filtrados**
Adobe	Contrato de suscripción a software como servicio (SaaS) de Adobe.	7	297	89	87
Anthropic	Acuerdo de Anthropic para servicios de IA ofrecidos como SaaS a través de AWS.	10	162	112	105
Atlassian	Acuerdo de Atlassian para productos en la nube (SaaS).	8	326	189	186
BlazeMeter	Acuerdo de Perforce para plataforma SaaS de pruebas de rendimiento (BlazeMeter).	14	401	177	173

La tabla continúa en la siguiente página

Continuación de la Tabla XVIII

Nombre	**Descripción**	**N° páginas**	**Tamaño (KB)**	**Total chunks**	**Chunks filtrados**
Canva	Contrato de Canva para su servicio de diseño SaaS, que regula el uso de su plataforma en la nube.	11	180	90	84
Figma	Acuerdo de Figma para su plataforma SaaS de diseño colaborativo, donde se establecen las condiciones de uso del servicio.	18	222	219	213
GitHub	Acuerdo de GitHub para productos en línea (SaaS), que define los términos asociados a su uso como servicio en la nube.	7	207	146	143
HubSpot	Términos de servicio de HubSpot para su plataforma SaaS.	40	599	424	418

La tabla continúa en la siguiente página

Continuación de la Tabla XVIII

Nombre	Descripción	N° páginas	Tamaño (KB)	Total chunks	Chunks filtrados
OpenAI	Acuerdo de OpenAI relacionado con servicios de inteligencia artificial en modalidad SaaS.	10	158	196	180
Oracle	Acuerdo de Oracle referente a servicios en la nube - Modalidad SaaS.	9	188	116	103

Los contratos del corpus presentan diferencias bastante marcadas en cuanto a extensión y complejidad, lo que, en el fondo, permite evaluar cómo se comporta el sistema frente a documentos de distinta naturaleza.

En cuanto al proceso, se aplica un chunking basado en similitud del coseno, con un umbral de 0.5 y sobre texto previamente limpio. Esto dio como resultado un promedio de 175.8 chunks por contrato. De estos, el filtro de ruido estructural descartó apenas un 3.75 % (aproximadamente 6.6 chunks), dejando disponibles para el análisis unos 169.2 chunks.

Este filtro descartó títulos cortos, encabezados con todo en mayúsculas, numeraciones aisladas y bloques con 70 % de sus caracteres en mayúsculas. Los chunks descartados son precisamente los elementos que, si se hubieran incorporado, habrían añadido ruido al proceso del análisis siguiente.

A modo ilustrativo, la Figura 12 muestra un fragmento del reporte HTML generado para uno de los contratos del corpus. El reporte identifica las cláusulas clasificadas por tipo (derechos,

obligaciones y prohibiciones) e indica para cada una el nivel de criticidad asignado por el agente analizador y la referencia a la sección específica del contrato de origen.

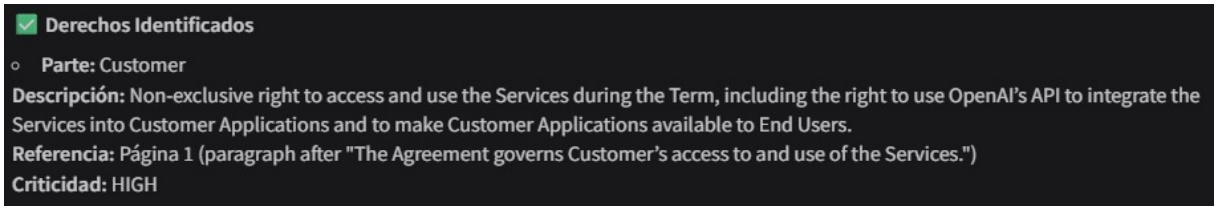


Figura 12.
Fragmento del reporte HTML con criticidad de cláusula

Durante las pruebas, se verificó el comportamiento del sistema frente a los casos borde definidos en el Sprint 08: contratos en formato PDF sin texto extraíble, archivos corruptos, pérdida de conexión con Qdrant y caídas del proveedor de modelos. En todos los escenarios, el sistema respondió con mensajes de error bastante descriptivos y, lo importante, sin interrumpir el proceso de forma abrupta, lo que en conjunto confirma la robustez del manejo de excepciones implementado.

4.3.2 Comparación con métodos manuales y otras herramientas

El proceso manual de revisión de los contratos SaaS puede convertirse en uno largo y duro, especialmente con respecto a la interpretación de términos especializados, cláusulas ambiguas o disposiciones que se afectan unas a otras a lo largo de todo el documento [3]. Así pues, el sistema diseñado no pretende reemplazar el juicio del abogado, sino aliviar su labor de repasar los documentos: identifica y clasifica automáticamente las cláusulas más relevantes, haciendo que se centre en lo más importante.

Para la evaluación objetiva de los análisis generados se utilizó el framework DeepEval, que implementa G-Eval como mecanismo de puntuación. Para Liu Y et al. [81], G-Eval es un marco que emplea un modelo de lenguaje como juez para evaluar la calidad de textos generados mediante criterios explícitos y pasos de razonamiento definidos.

Su principal ventaja frente a métricas basadas en coincidencia léxica, como BLEU o ROU-

GE, es que evalúa el significado y la pertinencia del contenido, no la similitud superficial de las palabras. Liu Y et al. [81], fundamentan G-Eval precisamente en su alta alineación con evaluaciones humanas, lo que lo convierte en un instrumento adecuado para validar los análisis jurídicos generados por el sistema.

Las cuatro métricas evaluadas mediante G-Eval corresponden a las definidas en la Tabla XIV del Capítulo 3. La primera, Fidelidad, verifica que las afirmaciones del análisis sean rastreables en el contrato original. La segunda, Relevancia, mide hasta qué punto la respuesta atiende la consulta del usuario.

Por su parte, la Precisión de Contexto evalúa qué proporción de los chunks recuperados resultan realmente pertinentes, mientras que Recall de Contexto cuantifica cuántos de los chunks relevantes disponibles fueron efectivamente recuperados. Juntas, estas métricas ofrecen una visión tanto de la calidad generativa del sistema como de su capacidad para recuperar información con precisión.

4.4 Análisis de Métricas

Analizar las métricas es observar a través de dos dimensiones interconectadas. Por un lado, la eficiencia de operación del sistema, registrando automáticamente en metrics.csv, y por otro, precisión semántica en los análisis generados, medida usando G-Eval y verificado por evaluación humana. El contraste de ambas perspectivas nos muestra que hace posible saber cuál es el equilibrio entre la calidad analítica y la rapidez en responder.

4.4.1 Rendimiento de los agentes

El rendimiento de los agentes constituye un indicador clave para evaluar la efectividad del sistema desarrollado. En esta sección se analizan los resultados obtenidos en cuanto a la capacidad de respuesta y comportamiento operativo de los agentes bajo distintas condiciones de ejecución, prestando especial atención a la eficiencia temporal como métrica principal para de-

terminar la viabilidad del sistema en entornos de uso real.

- **Eficiencia temporal de los agentes**

  La Tabla XIX presenta los estadísticos descriptivos calculados sobre los 10 contratos de prueba para cada tipo de operación, equipo y agente. Los campos registrados en metrics.csv (timestamp, agente, operación, documento, tiempo y estado), donde todos los valores de tiempo están expresados en segundos, permitieron calcular la media, la mediana, el valor mínimo, el valor máximo y la desviación estándar de la latencia por operación en cada condición del análisis comparativo.

Tabla XIX.

ESTADÍSTICOS DESCRIPTIVOS DE LATENCIA POR AGENTE, OPERACIÓN Y EQUIPO

Agente	Operación	N° Solicitudes	Lenovo (s)					Acer (s)				
			Prom.	Med.	Máx.	Mín.	D.E.	Prom.	Med.	Máx.	Mín.	D.E.
Orquestador	Almacena el análisis	30	25,23	14,35	96,46	1,96	30,94	15,32	4,01	213,57	1,97	38,17
Orquestador	Almacena el documento	10	47,67	19,84	139,20	2,83	49,73	200,55	6,41	906,85	2,87	324,92
Orquestador	Analiza el documento	30	67,43	5,73	301,96	2,23	96,61	79,29	4,28	458,61	2,25	221,53
Orquestador	Ver el análisis	10	15,82	3,35	87,11	2,56	28,03	28,92	4,27	96,69	2,76	36,48
Orquestador	Cuántos análisis están almacenados?	10	113,66	5,22	536,21	2,50	178,81	75,96	25,87	308,38	2,35	101,76
Orquestador	Cuántos documentos están almacenados?	10	119,52	88,40	411,90	3,52	139,08	45,19	13,23	168,50	2,94	61,13
Orquestador	Qué documentos tienen análisis?	10	7,81	3,13	27,41	2,60	10,01	22,81	4,26	102,87	2,49	40,18
Almacenador	Get analyzed docs	10	4,39	0,11	12,72	0,04	4,62	7,10	8,14	22,13	0,04	5,36
Almacenador	Get stats	10	4,04	0,77	11,25	0,16	4,78	0,35	0,10	2,67	0,04	0,82
Almacenador	Retrieve analysis	10	0,36	0,32	0,81	0,07	0,27	0,30	0,21	1,33	0,09	0,37
Almacenador	Store analysis	30	0,43	0,44	1,14	0,05	0,30	0,15	0,11	0,53	0,04	0,12
Almacenador	Store pdf	10	0,13	0,13	0,25	0,05	0,07	0,08	0,07	0,21	0,03	0,05
Analizador	Analyze contract	30	155,44	128,88	513,14	55,01	104,41	215,16	143,43	866,22	41,92	176,57

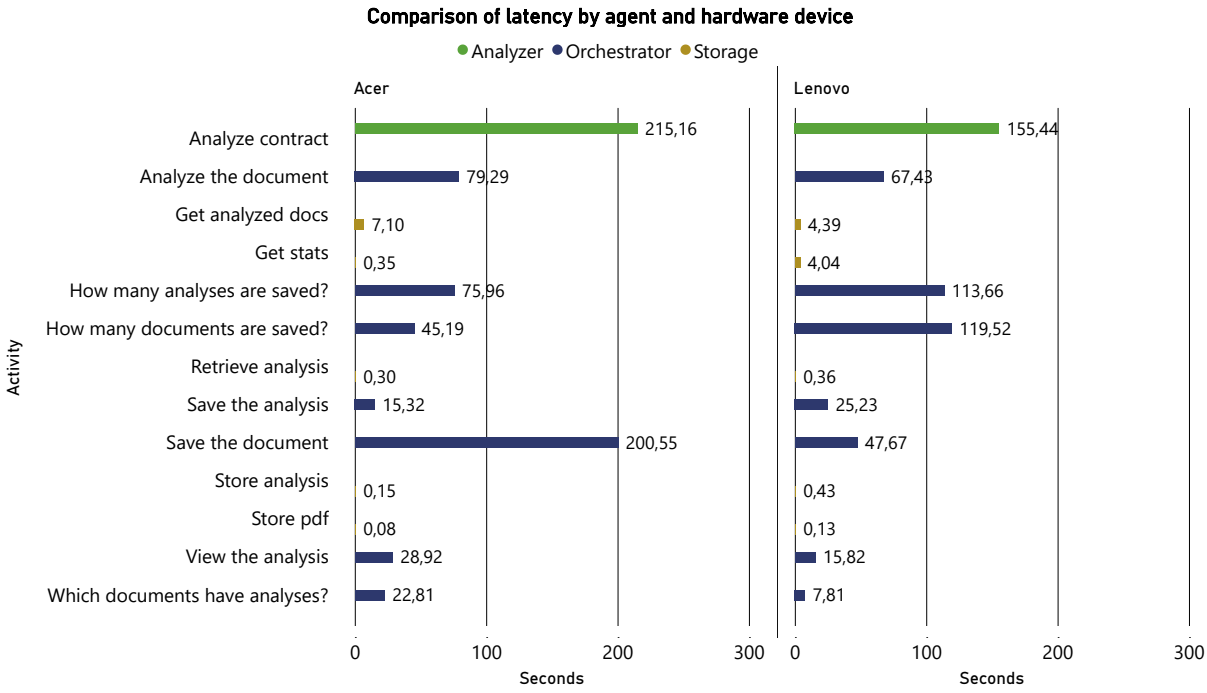


Figura 13.
Comparación de la latencia por agente y equipo de hardware

Los resultados muestran que la mayor parte del tiempo de procesamiento se concentra en el agente Analizador, donde el pipeline de CrewAI ejecuta de forma secuencial los dos sub-agentes. El agente Almacenador, cuya lógica de procesamiento se ejecuta mediante código estructurado en el executor sin intervención activa del LLM, presenta tiempos considerablemente menores y una variabilidad más reducida entre documentos como se muestra en la Figura 13.

Este comportamiento está alineado con lo previsto en el diseño del sistema: las operaciones de extracción de texto, chunking semántico y almacenamiento vectorial dependen directamente de la existencia de texto en el documento, es decir, de su contenido y extensión, más que del razonamiento del modelo en sí.

En cuanto a los componentes de hardware, se compara el equipo A (Lenovo) con el equipo B (Acer). Se observaron distintas diferencias en sus tiempos de ejecución. El Lenovo muestra menores latencias en las operaciones de mayor carga computacional. Un caso notable se halla en el agente Analizador: el Acer marcó un tiempo promedio de 215,16 s en comparación con 155,44 s para el Lenovo, una diferencia del 38,4 %. Por otro lado, la

desviación estándar del Acer durante esa operación llegó hasta los 176,57 s, lo que refleja una notable inestabilidad para ese entorno.

Incluso con eso, la ventaja del Lenovo no es sostenida a través de todo el sistema. En operaciones de menor carga como Almacena el análisis del Agente Orquestador, GetStatus y RetrieveAnalysis. El agente Almacenador del Acer registró tiempos promedio aún más bajos. Esto sugiere que el Lenovo funciona muy bien en el caso en que la carga computacional es alta, pero no de forma generalizada.

En conjunto, estos resultados sugieren que el hardware influye de manera más significativa en operaciones de mayor carga computacional, mientras que su impacto es reducido o incluso invertido en tareas más ligeras, manteniéndose en todos los casos el mismo patrón jerárquico de comportamiento entre agentes.

- **Rendimiento de los LLMs en el análisis de contratos por dispositivo**

  La Tabla XX muestra los estadísticos descriptivos (promedio (Prom), mediana (Med), mínimo (Mín), máximo (Máx) y desviación estándar (D.E.)) de los tiempos de procesamiento de los tres modelos LLM sobre el conjunto de 10 contratos en ambos dispositivos, considerando que todas las mediciones se encuentran expresadas en segundos.

Tabla XX.
MEDIDAS DESCRIPTIVAS DEL TIEMPO DE ANÁLISIS POR MODELO Y ENTORNO DE EJECUCIÓN

Modelo	**Lenovo**					**Acer**				
	Prom	**Med**	**Máx**	**Mín**	**D. E.**	**Prom**	**Med**	**Máx**	**Mín**	**D. E.**
Grok 4.1 Fast	70,89	69,91	83,87	55,01	9,70	111,36	112,05	162,42	88,25	20,83
Gemini 2.5 Flash Lite	161,73	155,84	259,98	79,52	63,99	175,30	185,39	286,46	41,92	94,35
Qwen3 VL 30B A3B Thinking	233,70	171,05	513,14	102,81	123,29	358,11	274,80	866,22	105,47	227,03

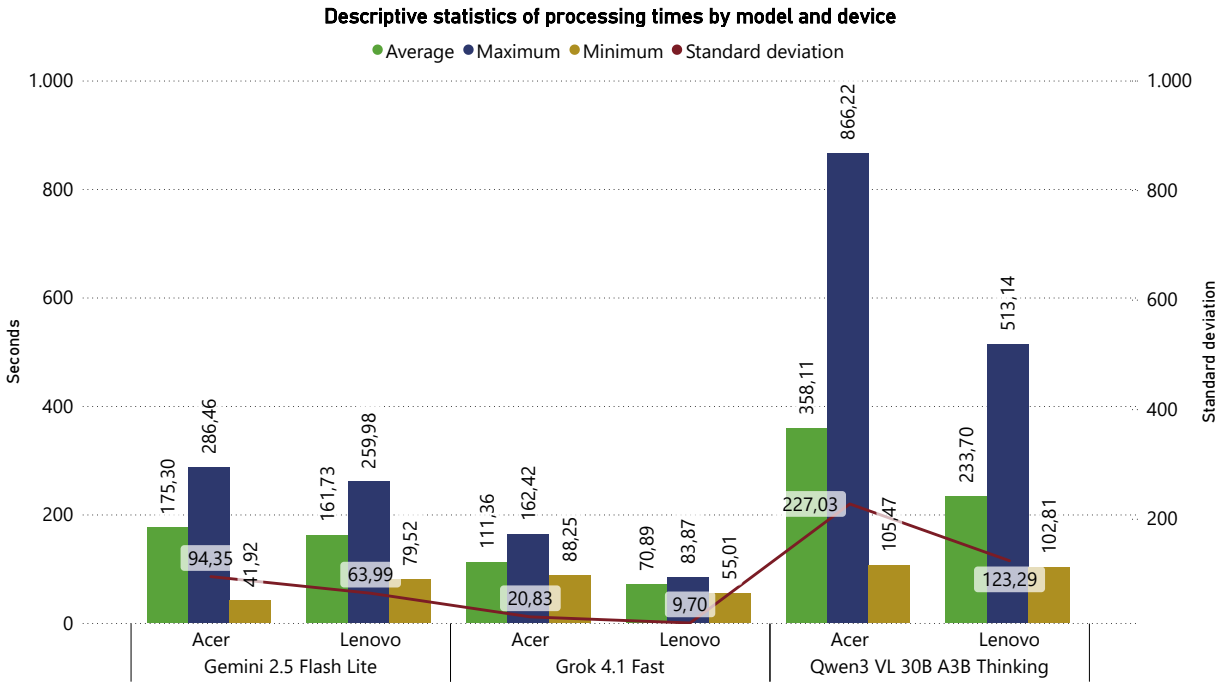


Figura 14.
Estadísticas descriptivas de los tiempos de procesamiento por modelo y dispositivo

- **Grok 4.1 Fast**

  Grok 4.1 Fast registra los tiempos de procesamiento más reducidos en el entorno Lenovo, con un promedio de 70,89 segundos y una mediana de 69,91 segundos, valores que evidencian una distribución temporal altamente equilibrada y, por tanto, un comportamiento consistente entre contratos.

  El tiempo mínimo fue de 55,01 segundos y el máximo de 83,87 segundos, un rango de variación de apenas 28,86 segundos. Con una desviación estándar de 9,70 segundos la más baja de todos los modelos en Lenovo Grok 4.1 Fast se posiciona como el modelo más estable en ese entorno.

  En el Acer, el promedio salta a los 111,36 segundos; en la mediana a 112,05 segundos, una diferencia con distribución igualmente equilibrada. Sin embargo, ahí la desviación estándar crece a 20,83 segundos. El rango se encuentra entre los 88,25 y 162,42 segundos. Es decir, cambiar de hardware puede agregar aún más variabilidad, pero Grok 4.1 Fast es todavía el modelo más rápido en promedio dentro del conjunto evaluado.

- **Gemini 2.5 Flash Lite**

  Gemini 2.5 Flash Lite muestra tiempos de procesamiento moderados frente al resto de modelos. En Lenovo, el promedio es de 161,73 segundos con una mediana de 155,84 segundos. La diferencia entre ambos valores apunta a un leve sesgo positivo: algunos contratos demandaron tiempos notablemente más altos que lo habitual. De hecho, el valor máximo alcanzó los 259,98 segundos, mientras que el mínimo fue de 79,52 segundos, lo que, en conjunto, deja un rango total de 180,46 segundos.

  La desviación estándar de 63,99 segundos es la más elevada del grupo de Lenovo, mostrando mucha variabilidad en el manejo del proceso. Esto podría explicarse como una diferencia estructural entre los contratos que comparamos o, sencillamente, porque es un comportamiento no determinista que se presenta en documentos de diferente complejidad.

  En Acer, el promedio se reduce ligeramente a 175,30 segundos con una mediana de 185,39 segundos, observándose en este caso un sesgo negativo que sugiere la presencia de contratos procesados con mayor rapidez que el valor central. La desviación estándar desciende a 94,35 segundos y el rango se extiende de 41,92 a 286,46 segundos, reflejando un comportamiento más disperso que en Lenovo, aunque con valores absolutos superiores a los de Grok en la misma plataforma.

- **Qwen3 VL 30B A3B Thinking**

  Qwen3 VL 30B A3B Thinking registra los tiempos de procesamiento más elevados del conjunto en ambas plataformas, lo cual es esperable dado que se trata del modelo de mayor número de parámetros activos evaluado.

  En Lenovo, el promedio alcanza los 233,70 segundos y la mediana se sitúa en 171,05 segundos. Esta diferencia de más de 62 segundos entre ambas medidas sugiere, en realidad, una distribución con sesgo positivo, arrastrada hacia arriba por valores extremos como el máximo de 513,14 segundos.

  Por otro lado, el mínimo de 102,81 segundos indica que este modelo es capaz de resolver contratos simples en tiempos bastante razonables. En cambio, la desviación estándar de 123,29 segundos, que es la más alta del conjunto, deja ver una cierta

inestabilidad que, en la práctica, afecta su consistencia operativa.

En el Acer los resultados cambian, aunque no necesariamente para mejor. El promedio sube a 358,11 segundos con una mediana de 274,80 segundos, y la desviación estándar de 227,03 segundos mantiene el mismo patrón de alta dispersión. El rango entre el mínimo de 105,47 y el máximo de 866,22 segundos es llamativo: sugiere que este modelo es especialmente sensible tanto al hardware como a la complejidad lingüística y estructural de cada contrato.

- **Comparativa General de Eficiencia Temporal**

  Desde una perspectiva comparativa, los datos permiten establecer una jerarquía clara en términos de eficiencia temporal como lo muestra la Figura 14: Grok 4.1 Fast se posiciona como el modelo más rápido y estable en ambas plataformas, seguido por Gemini 2.5 Flash Lite con tiempos intermedios pero mayor variabilidad, y finalmente Qwen3 VL 30B A3B Thinking como el modelo de mayor latencia y dispersión.

  Cabe señalar que el cambio de Lenovo a Acer aumenta los tiempos promedios para todos los modelos, algo sugerente de una penalización persistente de velocidad, aparentemente parte intrínseca de la arquitectura de Acer. Sin embargo, tal penalización no recae en todos los modelos. Gemini es quien más incrementa proporcionalmente su tasa de absorción del cambio de entorno. Qwen3 tiene la mayor sensibilidad al hardware. El primer punto también tiene relevancia en caso de planificar despliegues en infraestructura con computación limitada/variada.

- **Evaluación de la precisión semántica mediante G-Eval**

  La Tabla XXI presenta las puntuaciones promedio obtenidas con G-Eval para los tres modelos LLM evaluados sobre el corpus de 10 contratos. Los valores en bruto que dan origen a estas puntuaciones pueden consultarse en la Tabla XXIV (véase Anexos).

Tabla XXI.

PUNTUACIONES G-EVAL POR MODELO: FIDELIDAD (F), RELEVANCIA (AR), PRECISIÓN DE CONTEXTO (CP), RECALL DE CONTEXTO (CR)

Modelo	Lenovo				Acer			
	F	AR	CP	CR	F	AR	CP	CR
Grok 4.1 Fast	0,98	0	0,94	1	0,96	0,10	0,91	1
Gemini 2.5 Flash Lite	0,91	0,40	0,91	1	0,98	0,10	0,91	1
Qwen3 VL 30B A3B Thinking	0,95	0,20	0,90	1	0,96	0,40	0,91	1

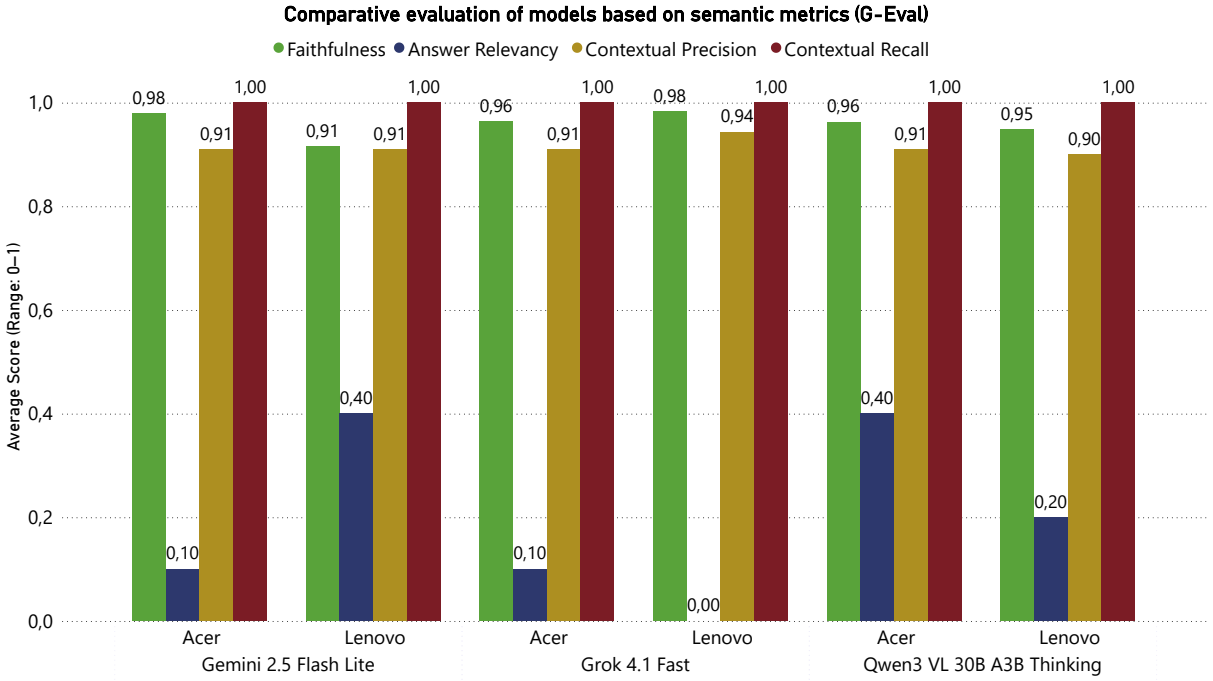


Figura 15.

Evaluación comparativa de modelos según métricas semánticas (G-Eval)

- **Grok 4.1 Fast**

  En términos de Fidelidad, Grok 4.1 Fast obtuvo la puntuación más alta entre todos los modelos evaluados en el dispositivo Lenovo (0,98), y una puntuación igualmente elevada en Acer (0,96), lo que indica que sus análisis contienen una mayor proporción de afirmaciones verificables en el texto contractual original.

  Este resultado sugiere que el modelo presenta menor tendencia a introducir infor-

mación no fundamentada en el documento, un aspecto crítico en el dominio jurídico donde la precisión factual tiene consecuencias directas sobre la interpretación de las cláusulas.

En cuanto a la relevancia, Grok 4.1 Fast registró los valores más bajos entre todos los modelos evaluados, con 0,00 en Lenovo y 0,10 en Acer. Esto, en la práctica, evidencia una dificultad bastante marcada para generar respuestas realmente pertinentes a las consultas planteadas, aunque, por otro lado, mantiene una alta fidelidad respecto al documento fuente.

En la precisión del contexto y el recall del contexto, los resultados son sólidos. Se obtienen en Lenovo valores de 0,94 y 1,00, y en Acer de 0,91 y 1,00. Un recall perfecto tanto en la plataforma uno como en la dos sugiere que el modelo recuperó toda su relevancia contextual en los casos. La precisión es un poco inferior a la obtenida para los valores de fidelidad, pero se mantiene en niveles altos y consistentes.

- **Gemini 2.5 Flash Lite**

  Con respecto a la fidelidad, Gemini 2.5 Flash Lite marcó 0,91 en Lenovo y 0,98 en Acer, el más alto de todos los modelos en la plataforma. Esto indica que sus afirmaciones están bien apoyadas en el contenido del contrato original, lo que sugiere que el modelo tiene una base factual sólida independientemente del entorno de ejecución y para las aplicaciones jurídicas en las que puede ser importante rastrear el origen de la información, esta consistencia es valiosa.

  En Relevancia, en cambio, los números cuentan otra historia. Gemini 2.5 Flash Lite obtuvo 0,40 en Lenovo y apenas 0,10 en Acer. Aunque supera a Grok en el entorno Lenovo, ambos valores siguen siendo bajos en términos absolutos, lo que apunta a dificultades reales para ajustar las respuestas a la intención específica de cada consulta sobre el corpus contractual.

  En lo concerniente a la precisión de contexto y el recall de contexto, en ambos casos se obtuvieron resultados de 0,91 y 1,00 respectivamente en ambas plataformas, por lo cual puede observarse una destacable estabilidad al recuperar el contexto relevante, logrando así consistentemente un recall perfecto con una precisión de contexto

uniforme.

- **Qwen3 VL 30B A3B Thinking**

  En relación a Fidelidad, Qwen3 VL 30B A3B Thinking obtuvo puntuaciones de 0,95 en Lenovo y 0,96 en Acer, valores que, aunque ligeramente inferiores a los líderes por métrica en cada plataforma, reflejan una fidelidad al texto contractual consistentemente alta y equilibrada entre entornos de ejecución.

  Este resultado indica que el modelo, aunque es de naturaleza open source, mantiene una proporción bastante sólida de afirmaciones verificables dentro del documento original, lo que, en conjunto, lo posiciona de forma competitiva frente a soluciones propietarias en el dominio jurídico.

  Por otro lado, en cuanto a la relevancia, Qwen3 alcanza valores de 0,20 en Lenovo y 0,40 en Acer. Este último, de hecho, es el valor más alto registrado hasta ahora dentro de las métricas de Acer, lo que sugiere que, bajo ciertas condiciones de hardware, la pertinencia de las respuestas mejora.

  En relación con la precisión del contexto y el recall del contexto, el modelo obtiene 0,90 y 1,00 en Lenovo, y 0,91 y 1,00 en Acer, manteniendo nuevamente un recall perfecto en ambas plataformas. A la par, la precisión del contexto se mantiene en niveles muy cercanos al modelo propietario, con apenas una diferencia marginal de 0,01 en Lenovo.

- **Comparativa: Modelos Propietarios vs. Modelo Open Source**

  Al contrastar el desempeño global de los modelos propietarios frente al modelo open source, se observa que las diferencias en las métricas principales son notablemente reducidas como se observa en la Figura 15. En Fidelidad, los tres modelos superan el umbral de 0,91 en ambas plataformas, lo que indica que ninguno introduce niveles significativos de alucinación factual en el contexto contractual.

  En el recall de contexto, los tres modelos obtienen un puntaje perfecto de 1,00. Esto ocurre por igual en modelos de pago y en Qwen3, lo que deja claro que recuperar el contexto completo no es un atributo exclusivo de los modelos comerciales. Donde sí aparece cierta

distancia es en la precisión del contexto: Grok obtuvo 0,94, mientras que Gemini obtuvo 0,91 en Lenovo, frente al 0,90 de Qwen3. Se observa una pequeña diferencia que, en todo caso, no modifica la utilidad del modelo open source.

La métrica que mayores diferencias presenta entre los modelos es la relevancia, donde ninguno de los tres alcanza valores realmente satisfactorios en términos absolutos. Esto, en el fondo, sugiere que la limitación es bastante transversal al tipo de licencia y que responde más bien a características del corpus o incluso a la propia metodología de evaluación.

En conclusión, Qwen3 VL 30B A3B Thinking demuestra un rendimiento bastante competitivo frente a los modelos propietarios evaluados, consolidándose como una alternativa viable en escenarios de análisis jurídico automatizado donde el costo de licenciamiento y la soberanía sobre los datos son factores importantes a tener en cuenta.

4.4.2 Satisfacción de usuarios piloto

La satisfacción de usuarios piloto es parte de una validación útil y aceptable del sistema por medio de un escenario realista. Mediante la evaluación humana, obtenemos la percepción directa de los usuarios acerca de la calidad, coherencia y utilidad de las respuestas proporcionadas por los agentes. Esto complementa a las medidas cuantitativas con un punto de vista subjetivo proveniente de una interacción real del sistema.

- **Evaluación humana**

  Con el objetivo de validar la confiabilidad de las puntuaciones obtenidas mediante G-Eval, se realizó una evaluación humana sobre el mismo conjunto de contratos de prueba. Para ello, se contó con dos evaluadores no especializados en el ámbito jurídico-contractual, quienes asumieron el rol de usuarios comunes.

  Cada evaluador calificó el análisis generado por los modelos LLM utilizando los cuatro criterios definidos por G-Eval: Fidelidad (F), Relevancia (AR), Precisión del Contexto (CP) y Recall del Contexto (CR). En este caso, se utilizó una escala Likert del 1 al 5 y,

posteriormente, los resultados se normalizaron al rango de 0 a 1, con el fin de que fueran comparables con las puntuaciones de G-Eval.

La Tabla XXII muestra las puntuaciones promedio derivadas de la evaluación humana aplicada a los tres modelos, considerando los criterios definidos sobre el conjunto de contratos. Los valores en bruto que dan origen a estas puntuaciones pueden consultarse en la Tabla XXV (véase Anexos).

Tabla XXII.
EVALUACIÓN HUMANA POR MODELO

Modelo	**Lenovo**				**Acer**			
	F	**AR**	**CP**	**CR**	**F**	**AR**	**CP**	**CR**
Grok 4.1 Fast	0,80	1	0,78	0,80	0,80	0,60	0,83	0,85
Gemini 2.5 Flash Lite	0,53	0,40	0,60	0,53	0,63	0,53	0,50	0,53
Qwen3 VL 30B A3B Thinking	0,65	0,50	0,65	0,75	0,73	0,58	0,68	0,65

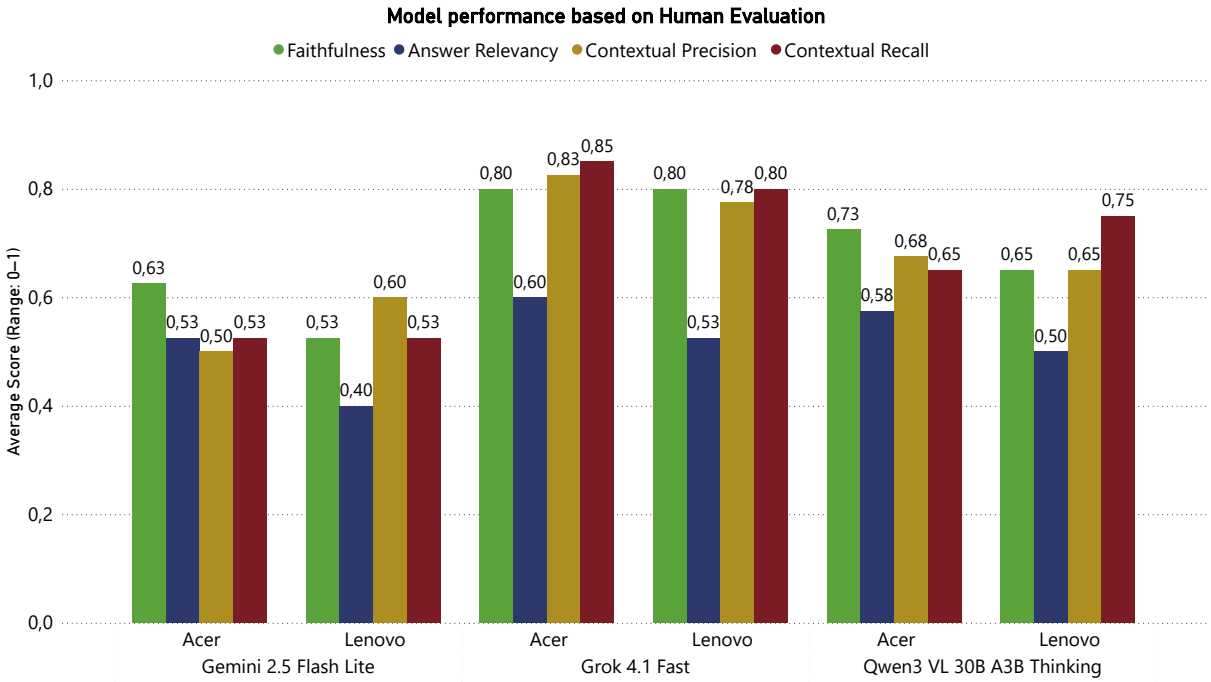


Figura 16.
Rendimiento del modelo basado en la evaluación humana

- **Grok 4.1 Fast**

  En relación con la Fidelidad, Grok 4.1 Fast obtuvo una puntuación de 0,80 tanto en Lenovo como en Acer, lo que refleja una percepción humana consistente y favorable respecto a la fidelidad de las respuestas generadas con relación al contenido contractual original. Esta estabilidad a través de las plataformas sugiere que la exactitud con respecto a la calidad del modelo no está relacionada con el entorno de ejecución, un comportamiento altamente deseado para aplicaciones legales, donde consistencia sobre exactitud del sistema es primario.

  Con respecto a la Relevancia, Grok 4.1 Fast registró el valor más alto del conjunto en Lenovo con una puntuación perfecta de 1,00, descendiendo a 0,60 en Acer, lo que sugiere que bajo el entorno Lenovo el modelo genera respuestas percibidas por el evaluador humano como plenamente pertinentes a las consultas contractuales planteadas, mientras que en Acer esta capacidad se ve moderadamente reducida.

  En cuanto a la precisión del contexto y el recall del contexto, el modelo obtuvo un puntaje de 0,78 y 0,80 respectivamente en Lenovo, y de 0,83 y 0,85 respectivamente en Acer; estos últimos son los valores más altos registrados en ambas métricas dentro de toda la tabla, lo que posiciona a Grok 4.1 Fast en los niveles superiores. Este modelo supera a los demás en cuanto a la recuperación precisa de contexto en la evaluación humana, particularmente en Acer.

- **Gemini 2.5 Flash Lite**

  En Fidelidad, Gemini 2.5 Flash Lite logró 0,53 en Lenovo y 0,63 en Acer, los valores más bajos del conjunto en esta métrica. Esto refleja que el evaluador humano encontró una parte considerable de afirmaciones pobremente respaldadas en el contrato original. Dentro del área legal, esto no es una cuestión menor, ya que introducir contenido complicado de verificar puede distorsionar la interpretación de las cláusulas y reducir la confiabilidad del sistema.

  En cuanto a la relevancia, los modelos obtuvieron 0,40 en Lenovo y 0,53 en Acer, valores que, en ambos casos, se mantuvieron por debajo del resto de modelos evaluados en las dos plataformas. Los evaluadores percibieron con menor frecuencia que

las respuestas estuvieran relacionadas con lo que se preguntaba. Una posible explicación es que el modelo tiende a incluir información contextual extra y, aunque sea relacionada, no siempre es estrictamente pertinente para la consulta en concreto.

En lo referente a Precisión de Contexto y Recall de Contexto, el modelo obtuvo los valores más bajos del conjunto en ambas plataformas: 0,60 y 0,53 en Lenovo, y 0,50 y 0,53 en Acer, evidenciando que tanto la capacidad para identificar los fragmentos contextualmente pertinentes como para recuperar la totalidad del contexto necesario resultan insuficientes bajo el criterio humano, con una ligera desmejora adicional al pasar al entorno Acer en Precisión de Contexto.

- **Qwen3 VL 30B A3B Thinking**

  En cuanto a la fidelidad, Qwen3 VL 30B A3B Thinking obtuvo 0,65 en Lenovo y 0,73 en Acer, situándose en valores intermedios entre Grok y Gemini en ambas plataformas. El incremento al pasar de un entorno al otro es bastante notable y sugiere que el modelo responde de forma favorable a las características del hardware Acer. En términos generales, mantiene una fidelidad factual aceptable según el criterio humano.

  Por otro lado, en relevancia, el modelo alcanzó 0,50 en Lenovo y 0,58 en Acer. Aunque no llega al nivel de Grok, sí supera a Gemini en ambas plataformas, lo que indica que tiene una capacidad razonable para generar respuestas que el evaluador experto considera pertinentes.

  En cuanto a la precisión del contexto y el recall del contexto, los resultados en Lenovo fueron 0,65 y 0,75, mientras que en Acer fueron 0,68 y 0,65 respectivamente. El recall de 0,75 en Lenovo representa el segundo valor más alto del conjunto en esta métrica, lo que indica que el modelo recupera una buena parte del contexto pertinente en ese entorno. En cambio, la disminución observada en Acer sugiere que esta capacidad no es del todo estable entre plataformas, lo cual, en la práctica, puede influir en su consistencia.

- **Comparativa General de Evaluación Humana**

  El análisis conjunto de las puntuaciones humanas permite establecer una jerarquía clara

entre los modelos evaluados como se observa en la Figura 16. Grok 4.1 Fast se consolida como el modelo con mejor desempeño global según el criterio humano, liderando en Fidelidad, Relevancia en Lenovo, y registrando los valores más altos de Precisión de Contexto y Recall en Acer.

Qwen3 VL 30B A3B Thinking ocupa una posición intermedia consistente, superando a Gemini en todas las métricas y plataformas, lo que refuerza su viabilidad como alternativa open source no solo desde la perspectiva automatizada sino también desde la validación por usuarios finales.

Gemini 2.5 Flash Lite, por su parte, registra las puntuaciones humanas más bajas del conjunto de forma bastante generalizada, lo que contrasta con su desempeño relativamente competitivo en las métricas automatizadas de G-Eval. Esto, en conjunto, sugiere que la evaluación humana pone en evidencia ciertas limitaciones en la calidad jurídica de sus respuestas que el juez LLM no logra captar con la misma sensibilidad.

Esta discrepancia entre ambos sistemas de evaluación es, probablemente, el hallazgo más significativo de la comparativa y vale la pena tenerla en cuenta al diseñar las próximas iteraciones del sistema.

- **Comparativa G-Eval vs. evaluación humana**

  La Tabla XXIII presenta las puntuaciones promedio de la evaluación humana junto con las obtenidas mediante G-Eval para cada modelo, permitiendo una comparación directa entre el juicio automatizado y el criterio humano. Los valores en bruto que dan origen a estas puntuaciones pueden consultarse en la Tabla XXVI (véase Anexos).

Tabla XXIII.

COMPARATIVA G-EVAL VS. EVALUACIÓN HUMANA POR MODELO Y MÉTRICA

Modelo	Equipo	G-Eval				Evaluación Humana			
		F	AR	CP	CR	F	AR	CP	CR
Grok 4.1 Fast	Lenovo	0,98	0	0,94	1	0,80	1	0,78	0,80
	Acer	0,96	0,10	0,91	1	0,80	0,60	0,83	0,85
Gemini 2.5 Flash Lite	Lenovo	0,91	0,40	0,91	1	0,53	0,40	0,60	0,53
	Acer	0,98	0,10	0,91	1	0,63	0,53	0,50	0,53
Qwen3 VL 30B A3B Thinking	Lenovo	0,95	0,20	0,90	1	0,65	0,50	0,65	0,75
	Acer	0,96	0,40	0,91	1	0,73	0,58	0,68	0,65

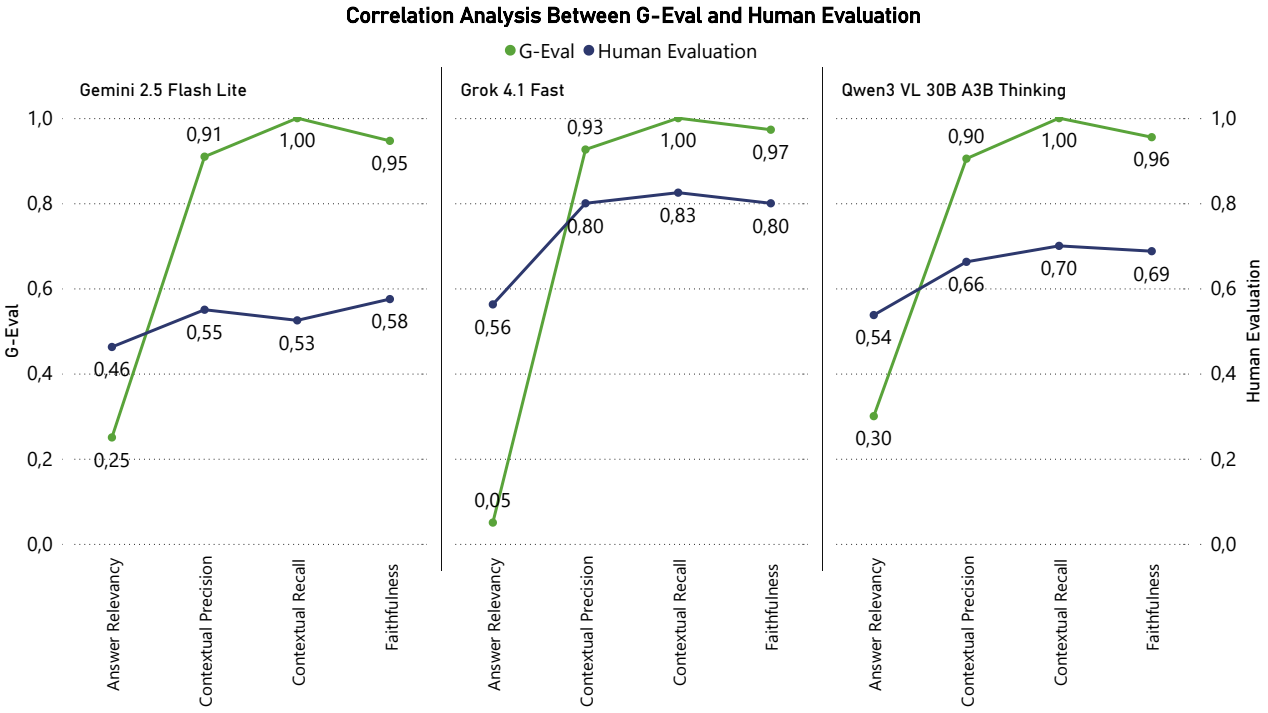


Figura 17.

Comparativa G-Eval vs. evaluación humana por modelo y métrica

El análisis de correlación entre las puntuaciones de G-Eval y las calificaciones humanas muestra una correlación moderada-alta en las métricas de Fidelidad y Recall de Contexto, pero una divergencia considerable en la Relevancia y Precisión de Contexto como se observa en la Figura 17.

En cuanto a la Fidelidad, los tres modelos presentan resultados superiores a 0,91 en ambas plataformas según la evaluación automatizada. En cambio, la evaluación humana los sitúa en un rango más bajo, entre 0,53 y 0,80, lo que sugiere que el juez LLM tiende a ser más generoso al valorar la fiabilidad factual que un evaluador humano.

Por otro lado, en el Recall de Contexto, G-Eval asigna una puntuación perfecta de 1,00 a los tres modelos, indicando que, bajo este criterio automatizado, se comportan de forma equivalente. Sin embargo, la evaluación humana muestra valores entre 0,53 y 0,85, lo que evidencia que el criterio humano es más sensible a diferencias en la cobertura del contexto recuperado y logra distinguirlas con mayor precisión.

Este resultado es consistente con lo reportado por Liu Y et al. [81], quienes fundamentan G-Eval en su alta alineación con evaluaciones humanas, lo que respalda la confiabilidad de las métricas automatizadas como instrumento de evaluación en el contexto de este proyecto, si bien los presentes datos invitan a interpretar dicha alineación como una aproximación robusta y no como una equivalencia exacta.

En cuanto a la Relevancia, Grok 4.1 Fast muestra la mayor diferencia entre ambos sistemas de evaluación. Por un lado, G-Eval le asigna 0,00 en Lenovo y 0,10 en Acer; sin embargo, la evaluación humana le otorga 1,00 en Lenovo y 0,60 en Acer, respectivamente.

En este caso se observa prácticamente una inversión completa del desempeño entre ambos enfoques, lo que, en conjunto, refleja una brecha bastante significativa en esta métrica.

Puede existir una explicación para esta diferencia. El evaluador humano probablemente valoró la relevancia temática general de la respuesta frente a la consulta contractual, mientras que el sistema automatizado aplicado fue más estricto al evaluar la alineación conceptual directa entre la pregunta y cada fragmento de la respuesta. Se trata de dos formas distintas de interpretar lo que significa una respuesta adecuada.

Por otro lado, las ambigüedades propias de ciertas cláusulas contractuales en el corpus pueden haber influido en la forma en que cada sistema interpreta qué constituye una respuesta relevante. Este conjunto reducido de contratos le da a dicha variabilidad un peso mayor del que tendría en un corpus más amplio.

Nada de esto invalida los resultados automatizados; al contrario, estas divergencias más bien señalan oportunidades de ajuste en el sistema para futuras iteraciones, especialmente en lo relacionado con el criterio de relevancia contextual que aplica el juez LLM en dominios jurídicos especializados.

- **Balance general**

  Tomando en conjunto las métricas de precisión semántica, la validación humana y la eficiencia temporal, Grok 4.1 Fast se perfila como la opción más equilibrada en el contexto de este proyecto.

  Registra la fidelidad más alta en Lenovo (0,98), además de los valores humanos más altos del grupo en esta métrica (0,80 en ambas plataformas). Su recall de contexto perfecto fue confirmado tanto por la evaluación automatizada como por el criterio humano en su mejor configuración (0,85 en Acer), mientras que su precisión de contexto automatizada también destaca como la más elevada del conjunto (0,94 en Lenovo).

  En contraste, su relevancia automatizada es la más baja; sin embargo, la evaluación humana invierte esta lectura. Los evaluadores perciben el modelo como más pertinente, lo que sugiere que sus respuestas son temáticamente adecuadas para el dominio jurídico, incluso si el juez LLM no logra reflejarlo con la misma precisión.

  Por otro lado, Gemini 2.5 Flash Lite logra valores de fidelidad comparables en la evaluación automatizada, alcanzando 0,98 en Acer e igualando a Grok en este aspecto. No obstante, sus resultados en la evaluación humana son consistentemente inferiores en todas las métricas, con rangos de 0,40 a 0,63 en fidelidad y de 0,50 a 0,53 tanto en precisión como en recall de contexto.

  En conjunto, se trata de un resultado sólido desde el punto de vista automatizado, pero que no se sostiene completamente frente a la evaluación humana. Aun así, su menor latencia puede ser razonable en escenarios donde se prioriza la velocidad por encima de una validación humana más consistente.

  Por su parte, Qwen3 VL 30B A3B Thinking sorprende. Sus métricas automatizadas son comparables a las de los modelos propietarios en Fidelidad (0,95–0,96) y Recall del Con-

texto. La evaluación humana reporta valores de Fidelidad entre 0,65 y 0,73, y de Precisión del Contexto entre 0,65 y 0,68. Esto lo sitúa por encima de Gemini en diversas configuraciones. Para ser un modelo open source, los resultados son más que competitivos.

Eso, sumado a su naturaleza de código abierto, lo convierte en una alternativa real para implementaciones que prioricen consistencia e independencia de proveedores comerciales. Especialmente en contextos institucionales o académicos donde el control sobre los datos, la auditabilidad del sistema y la ausencia de costos de licenciamiento no son opcionales.

CONCLUSIONES

El sistema desarrollado demostró ser capaz de automatizar de forma efectiva la extracción, almacenamiento y análisis de cláusulas en contratos SaaS mediante una arquitectura multiagente coordinada por el protocolo A2A. La distribución de responsabilidades entre tres agentes especializados, orquestador, almacenador y analizador, permitió estructurar el procesamiento de manera organizada y predecible, reduciendo la complejidad propia del análisis de documentos jurídicos extensos.

La estrategia de chunking semántico basada en similitud del coseno fue exitosa para mantener la consistencia semántica en los fragmentos contractuales. En promedio, produjo 175,8 chunks por contrato, y en su mayoría (96,25 %) estos pasaron el filtro de ruido estructural. Los tres modelos analizados obtuvieron resultados de Fidelidad por encima de 0,91 y un Recall de Contexto perfecto (1,00) en ambas plataformas; estos resultados confirman el alto porcentaje de afirmaciones verificables contenidas en las respuestas y la capacidad del sistema para recuperar todo el contexto relevante.

El modelo Grok 4.1 Fast logró un buen equilibrio entre la calidad del análisis y la eficiencia en tiempo. Registró la Fidelidad automatizada más alta en Lenovo (0,98) y, además, presentó el menor tiempo promedio del conjunto (70,89 segundos), junto con la desviación estándar más baja (9,70 segundos), lo que indica un comportamiento bastante estable.

Por otro lado, la evaluación humana también respalda estos resultados: fue el modelo con mayor pertinencia percibida en Relevancia (1,00 en Lenovo), así como con los valores más altos de Precisión de Contexto y Recall en Acer (0,83 y 0,85, respectivamente).

El trabajo aporta evidencia empírica sobre la viabilidad de aplicar arquitecturas multiagente basadas en el protocolo A2A al análisis automatizado de contratos SaaS sin recurrir a procesos de fine-tuning, apoyándose en las capacidades de comprensión jurídica adquiridas por los modelos durante su preentrenamiento, en línea con lo establecido por Zin et al. [14].

La comparativa entre modelos propietarios y el modelo open source Qwen3 VL 30B A3B

Thinking representa una contribución bastante relevante. Sus métricas automatizadas resultan comparables a las de los modelos propietarios, con diferencias mínimas en Fidelidad y Precisión de Contexto, lo que, en conjunto, lo posiciona como una alternativa viable para implementaciones donde se prioriza la independencia de proveedores comerciales y la soberanía sobre los datos.

Por otro lado, el análisis de correlación entre G-Eval y la evaluación humana muestra una correspondencia de nivel modesto-alto en Fidelidad y Recall del Contexto. Sin embargo, en Relevancia las diferencias son más marcadas, especialmente en el caso de Grok 4.1 Fast: G-Eval le asigna 0,00 en Lenovo, mientras que el evaluador humano le otorga 1,00.

Esta brecha sugiere que el juez LLM aplica un criterio de relevancia más estricto que el de un evaluador humano y, en consecuencia, parece necesario ajustar los parámetros de la evaluación automatizada considerando el dominio jurídico y el tipo de usuario del sistema

RECOMENDACIONES

La arquitectura modular del sistema y el uso de modelos de lenguaje de propósito general permiten extender su aplicabilidad a otros tipos de documentos contractuales como acuerdos de confidencialidad, contratos de prestación de servicios y licenciamiento de software.

Para ello se recomienda revisar el diseño de los prompts del agente analizador según las categorías de cláusulas propias de cada dominio, y ampliar el corpus de prueba incorporando contratos de mayor variedad en cuanto a jurisdicciones e industrias, con el fin de evaluar la robustez del sistema ante estructuras contractuales más diversas.

Para futuras evaluaciones, vale la pena considerar modelos de código abierto más recientes, prestando atención a los que cuenten con mayor ventana de contexto o especialización en dominios legales. También sería interesante explorar técnicas de fine-tuning sobre corpus jurídicos anotados, con el objetivo de determinar si ajustar el modelo al dominio contractual específico logra mejorar las métricas de Relevancia, que en este trabajo registraron los valores más bajos del conjunto en la evaluación automatizada.

El corpus de evaluación está compuesto por 10 contratos SaaS reales. En conjunto, cubre un espectro algo limitado de jurisdicciones y tipos de servicio, lo que, en la práctica, dificulta la generalización de los resultados hacia otros contextos regulatorios distintos. Por eso, en estudios futuros, un corpus más amplio con mayor variedad de contratos y jurisdicciones ayudaría a reforzar la validez y generalización de los hallazgos.

Por otro lado, la evaluación humana aportó una referencia útil como validación externa, aunque los evaluadores no contaban con especialización en el ámbito jurídico-contractual. Esto, de alguna forma, limita la profundidad con la que se aplican criterios como Fidelidad o Precisión del Contexto. En ese sentido, contar con abogados especialistas en futuros estudios permitiría una validación más sólida y representativa de la calidad jurídica del análisis.

Finalmente, el sistema se apoya en OpenRouter como punto de conexión unificado para los modelos de lenguaje, lo que implica la existencia de un punto único de fallo externo. Para reducir

este riesgo en versiones futuras, sería recomendable explorar una integración más directa con las APIs individuales de cada proveedor, así como alternativas de respaldo, como la implementación de modelos ejecutados localmente para mantener la disponibilidad del sistema ante posibles interrupciones del servicio.

BIBLIOGRAFÍA

[1] H. W. Classen, *Acuerdos SaaS: Disposiciones contractuales clave*, nov. de 2021. dirección: https://www.americanbar.org/groups/business_law/resources/business-law-today/2021-november/saas-agreements-key-contractual-provisions/.

[2] Kunzapp, *Cómo navegar los contratos SaaS*, ene. de 2025. dirección: https://www.linkedin.com/pulse/c%C3%B3mo-navegar-los-contratos-saas-kunzapp-szaff/.

[3] A. Seabra, C. Cavalcante, J. Nepomuceno, L. Lago, N. Ruberg y S. Lifschitz, *Contrato360 2.0: A Document and Database-Driven Question-Answer System using Large Language Models and Agents*, dic. de 2024. dirección: https://arxiv.org/pdf/2412.17942.

[4] M. Gutiérrez et al., "Aprendizaje Cooperativo Multiagente," *Revista Electro*, vol. 44, págs. 614-201, oct. de 2022, issn: 1405-2172. dirección: http://electro.itchihuahua.edu.mx/revista.

[5] P. Raptopoulos, G. Filandrianos, M. Lymperaiou y G. Stamou, *PAKTON: A Multi-Agent Framework for Question Answering in Long Legal Agreements*, sep. de 2025. dirección: http://arxiv.org/abs/2506.00608.

[6] C. C. Liao, D. Liao y S. S. Gadiraju, *AgentMaster: A Multi-Agent Conversational Framework Using A2A and MCP Protocols for Multimodal Information Retrieval and Analysis*, jul. de 2025. dirección: https://arxiv.org/pdf/2507.21105.

[7] P. Pajo, *Comprehensive Analysis of Google's Agent2Agent (A2A) Protocol: Technical Architecture, Enterprise Use Cases, and Long-Term Implications for AI Collaboration*, abr. de 2025. doi: 10.13140/RG.2.2.32649.56164.

[8] Google for Developers, *Announcing the Agent2Agent Protocol (A2A)*, abr. de 2025. dirección: https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/.

[9] Y. Yang et al., *A Survey of AI Agent Protocols*, jun. de 2025. arXiv: 2504.16736 [cs.AI]. dirección: https://arxiv.org/abs/2504.16736.

[10] S. Joshi, "Review of Autonomous and Collaborative Agentic AI and Multi-Agent Systems for Enterprise Applications," *International Journal of Innovative Research in Engineering and Management*, vol. 12, págs. 65-76, 3 jul. de 2025. doi: 10.55524/ijirem.2025.12.3.9.

[11] J. Breton et al., "Leveraging LLMs for legal terms extraction with limited annotated data," *Artificial Intelligence and Law*, feb. de 2025, issn: 15728382. doi: 10.1007/S10506-025-09448-8.

[12] H. Derouiche, Z. Brahmi y H. Mazeni, *Agentic AI Frameworks: Architectures, Protocols, and Design Challenges*, 2025. arXiv: 2508.10146 [cs.AI]. dirección: https://arxiv.org/abs/2508.10146.

[13] M. A. Ferrag, N. Tihanyi y M. Debbah, *From LLM Reasoning to Autonomous AI Agents: A Comprehensive Review*, 2025. arXiv: 2504.19678 [cs.AI]. dirección: https://arxiv.org/abs/2504.19678.

[14] M. M. Zin, H. T. Nguyen, K. Satoh, S. Sugawara y F. Nishino, "Information Extraction from Lengthy Legal Contracts: Leveraging Query-Based Summarization and GPT-3.5," *Frontiers in Artificial Intelligence and Applications*, vol. 379, págs. 177-186, dic. de 2023, issn: 18798314. doi: 10.3233/FAIA230963.

[15] E. W. Kim, Y. J. Shin, K. J. Kim y S. Kwon, "Development of an Automated Construction Contract Review Framework Using Large Language Model and Domain Knowledge," *Buildings*, vol. 15, n.º 6, mar. de 2025, issn: 2075-5309. doi: 10.3390/BUILDINGS15060923.

[16] B. Lundell, J. Gamalielsson, A. Katz y M. Lindroth, "Use of Commercial SaaS Solutions in Swedish Public Sector Organisations under Unknown Contract Terms," en *Electronic Government*, M. Janssen et al., eds., ép. Lecture Notes in Computer Science, vol. 13391, Cham: Springer International Publishing, 2022, págs. 73-92, isbn: 978-3-031-15086-9. doi: 10.1007/978-3-031-15086-9_6.

[17] T. Lynn, “Dear Cloud, I Think We Have Trust Issues: Cloud Computing Contracts and Trust,” *Palgrave Studies in Digital Business and Enabling Technologies*, págs. 21-42, 2021, Cited by: 5; All Open Access, Gold Open Access. doi: 10.1007/978-3-030-54660-1_2.

[18] S. Minaee et al., *Large Language Models: A Survey*, 2025. arXiv: 2402.06196 [cs.CL]. dirección: https://arxiv.org/abs/2402.06196.

[19] Z. Liu et al., *LLM360: Towards Fully Transparent Open-Source LLMs*, 2023. arXiv: 2312.06550 [cs.CL]. dirección: https://arxiv.org/abs/2312.06550.

[20] H. Touvron et al., *Llama 2: Open Foundation and Fine-Tuned Chat Models*, 2023. arXiv: 2307.09288 [cs.CL]. dirección: https://arxiv.org/abs/2307.09288.

[21] A. Q. Jiang et al., *Mistral 7B*, 2023. arXiv: 2310.06825 [cs.CL]. dirección: https://arxiv.org/abs/2310.06825.

[22] E. Almazrouei et al., *The Falcon Series of Open Language Models*, 2023. arXiv: 2311.16867 [cs.CL]. dirección: https://arxiv.org/abs/2311.16867.

[23] O. A. y Enis Karaarslan y Fatih Safa Erenay y Neboja Baanin, *IA generativa en la escritura académica: una comparación de DeepSeek, Qwen, ChatGPT, Gemini, Llama, Mistral y Gemma*, 2025. arXiv: 2503.04765 [cs.CY]. dirección: https://arxiv.org/abs/2503.04765.

[24] N. P. P. Juttu, S. Singireddy, S. Gona y S. Timilsina, *Text to Trust: Evaluating Fine-Tuning and LoRA Trade-offs in Language Models for Unfair Terms of Service Detection*, 2025. arXiv: 2510.22531 [cs.CL]. dirección: https://arxiv.org/abs/2510.22531.

[25] X.-K. Wu et al., “LLM Fine-Tuning: Concepts, Opportunities, and Challenges,” *Big Data and Cognitive Computing*, vol. 9, n.º 4, 2025, issn: 2504-2289. doi: 10.3390/bdcc9040087.

[26] S. Klem y N. A. Moubayed, *LLMs for LLMs: A Structured Prompting Methodology for Long Legal Documents*, 2025. arXiv: 2509.02241 [cs.AI]. dirección: https://arxiv.org/abs/2509.02241.

[27] D. Trautmann, A. Petrova y F. Schilder, *Legal Prompt Engineering for Multilingual Legal Judgement Prediction*, 2022. arXiv: 2212.02199 [cs.CL]. dirección: https://arxiv.org/abs/2212.02199.

[28] F. Yu, L. Quartey y F. Schilder, *Exploring the Effectiveness of Prompt Engineering for Legal Reasoning Tasks*, A. Rogers, J. Boyd-Graber y N. Okazaki, eds., Toronto, Canada, jul. de 2023. doi: 10.18653/v1/2023.findings-acl.858.

[29] D. Premasiri, T. Ranasinghe, R. Mitkov, M. El-Haj e I. Frommholz, "Survey on legal information extraction: current status and open challenges," *Knowledge and Information Systems*, 2025, Cited by: 0; All Open Access, Hybrid Gold Open Access. doi: 10.1007/s10115-025-02600-5.

[30] M. M. Rahman, N. G. Md, M. M. Rahman, M. M. Rahman y M. K. S. Sd, "Natural language processing in legal document analysis software: A systematic review of current approaches, challenges, and opportunities," *International Journal of Innovative Research and Scientific Studies*, 2025. doi: DOI:10.53894/ijirss.v8i3.7702.

[31] W. Xu, Y. Deng, W. Lei, W. Zhao, T.-S. Chua y W. Lam, *ConReader: Exploring Implicit Relations in Contracts for Contract Clause Extraction*, Y. Goldberg, Z. Kozareva e Y. Zhang, eds., Abu Dhabi, United Arab Emirates, dic. de 2022. doi: 10.18653/v1/2022.emnlp-main.166.

[32] C. Eppright, *¿Qué es el procesamiento de lenguaje natural (NLP)?* Mar. de 2021. dirección: https://www.oracle.com/es/artificial-intelligence/natural-language-processing/.

[33] K. Luo, Q. Huang, C. Jiang e Y. Feng, *Automating Legal Interpretation with LLMs: Retrieval, Generation, and Evaluation*, W. Che, J. Nabende, E. Shutova y M. T. Pilehvar, eds., Vienna, Austria, jul. de 2025. doi: 10.18653/v1/2025.acl-long.204.

[34] J. Valvoda, A. Thompson, R. Cotterell y S. Teufel, "The Ethics of Automating Legal Actors," *Transactions of the Association for Computational Linguistics*, vol. 12, págs. 700-720, 2024, Cited by: 3; All Open Access, Gold Open Access. doi: 10.1162/tacl_a_00668.

[35] K.-T. Tran, D. Dao, M.-D. Nguyen, Q.-V. Pham, B. O'Sullivan y H. D. Nguyen, *Multi-Agent Collaboration Mechanisms: A Survey of LLMs*, 2025. arXiv: 2501.06322 [cs.AI]. dirección: https://arxiv.org/abs/2501.06322.

[36] X. Li, S. Wang, S. Zeng, Y. Wu e Y. Yang, "A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges," *Vicinagearth*, 2024. doi: 10.1007/s44336-024-00009-2.

[37] B. Yan et al., *Beyond Self-Talk: A Communication-Centric Survey of LLM-Based Multi-Agent Systems*, 2025. arXiv: 2502.14321 [cs.MA]. dirección: https://arxiv.org/abs/2502.14321.

[38] P. Shi y B. Yan, "A Survey on Intelligent Control for Multiagent Systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, n.º 1, págs. 161-175, 2021. doi: 10.1109/TSMC.2020.3042823.

[39] Y. Zhao, W. Shi, F. Feng y X. He, *AppAgent-Pro: A Proactive GUI Agent System for Multidomain Information Integration and User Assistance*, 2025. arXiv: 2508.18689 [cs.AI]. dirección: https://arxiv.org/abs/2508.18689.

[40] A. Ehtesham, A. Singh, G. K. Gupta y S. Kumar, *A survey of agent interoperability protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP)*, 2025. arXiv: 2505.02279 [cs.AI]. dirección: https://arxiv.org/abs/2505.02279.

[41] I. Habler, K. Huang, V. S. Narajala y P. Kulkarni, *Building A Secure Agentic AI Application Leveraging A2A Protocol*, 2025. arXiv: 2504.16902 [cs.CR]. dirección: https://arxiv.org/abs/2504.16902.

[42] P. Weber, *What is an Agent Communication Language?* Nov. de 2024. dirección: https://smythos.com/developers/agent-development/agent-communication-language/.

[43] X. Luo, Y. Li, Q. Huang y J. Zhan, "A survey of automated negotiation: Human factor, learning, and application," *Computer Science Review*, vol. 54, pág. 100 683, 2024, issn: 1574-0137. doi: https://doi.org/10.1016/j.cosrev.2024.100683.

[44] Y. Zhang, J. Wu y R. Cao, "Optimizing Automated Negotiation: Integrating Opponent Modeling with Reinforcement Learning for Strategy Enhancement," *Mathematics*, vol. 13, n.º 4, 2025, issn: 2227-7390. doi: 10.3390/math13040679.

[45] M. O. Keskin, F. Cantürk, C. Eran y R. Aydoğan, "Decentralized multi-agent path finding framework and strategies based on automated negotiation," *Autonomous Agents and Multi-Agent Systems*, vol. 38, 1 jun. de 2024, issn: 15737454. doi: 10.1007/s10458-024-09639-8.

[46] J. Anglen, *Multi-Agent Environment Tools: Top Frameworks*, jun. de 2025. dirección: https://www.rapidinnovation.io/post/frameworks-and-tools-for-building-multi-agent-environments.

[47] Z. Wrona et al., "Comparison of Multi-Agent Platform Usability for Industrial-Grade Applications," *Applied Sciences*, vol. 14, n.º 22, 2024, issn: 2076-3417. doi: 10.3390/app142210124.

[48] F. Ariai, J. Mackenzie y G. Demartini, *Natural Language Processing for the Legal Domain: A Survey of Tasks, Datasets, Models, and Challenges*, 2025. arXiv: 2410.21306 [cs.CL]. dirección: https://arxiv.org/abs/2410.21306.

[49] S. K. Nigam, T. Dubey, G. Sharma, N. Shallum, K. Ghosh y A. Bhattacharya, *LegalSeg: Unlocking the Structure of Indian Legal Judgments Through Rhetorical Role Classification*, 2025. arXiv: 2502.05836 [cs.CL]. dirección: https://arxiv.org/abs/2502.05836.

[50] J. A. de Oliveira Lima, *Unlocking Legal Knowledge with Multi-Layered Embedding-Based Retrieval*, 2024. arXiv: 2411.07739 [cs.AI]. dirección: https://arxiv.org/abs/2411.07739.

[51] A. Brown, M. Roman y B. Devereux, *A Systematic Literature Review of Retrieval-Augmented Generation: Techniques, Metrics, and Challenges*, 2025. arXiv: 2508.06401 [cs.DL]. dirección: https://arxiv.org/abs/2508.06401.

[52] L. Monigatti y Z. Hasan, *A Gentle Introduction to Vector Databases*, ago. de 2023. dirección: https://weaviate.io/blog/what-is-a-vector-database.

[53] Weaviate, *Weaviate Repository*, dic. de 2025. dirección: https://github.com/weaviate/weaviate.

[54] A. Ricadela, *¿Qué es Weaviate?* Mar. de 2025. dirección: https://www.oracle.com/latam/database/vector-database/weaviate/.

[55] Chroma, *Chroma Introduction*, 2025. dirección: https://docs.trychroma.com/docs/overview/introduction.

[56] IONOS, *¿Qué es Chroma DB?* Mar. de 2025. dirección: https://www.ionos.com/es-us/digitalguide/servidores/know-how/chroma-db/.

[57] M. Kosinski, *¿Qué es Milvus?* Dic. de 2025. dirección: https://www.ibm.com/mx-es/think/topics/milvus.

[58] milvus, *¿Qué es Milvus?* Dic. de 2025. dirección: https://milvus.io/docs/es/overview.md.

[59] S. Dua, *¿Qué es Qdrant?* Jul. de 2025. dirección: https://www.oracle.com/latam/database/vector-database/qdrant/.

[60] V. Botti, *Agentic AI and Multiagentic: Are We Reinventing the Wheel?* 2025. arXiv: 2506.01463 [cs.MA]. dirección: https://arxiv.org/abs/2506.01463.

[61] Q. Wu et al., *AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation*, 2023. arXiv: 2308.08155 [cs.AI]. dirección: https://arxiv.org/abs/2308.08155.

[62] Haystack, *Introduction to Haystack*, nov. de 2025. dirección: https://docs.haystack.deepset.ai/docs/intr.

[63] CrewAI, *What is CrewAI?* Nov. de 2025. dirección: https://docs.crewai.com/en/introduction.

[64] B. Bednarz y M. Miłosz, "Benchmarking the performance of Python web frameworks," *Journal of Computer Sciences Institute*, vol. 36, págs. 336-341, 2025. doi: 10.35784/jcsi.7738.

[65] T. Ge, Y. Liu, J. Ye, T. Li y C. Wang, *Advancing vision-language models in front-end development via data synthesis*, 2025. arXiv: 2503.01619 [cs.CV]. dirección: https://arxiv.org/abs/2503.01619.

[66] H. Q. Zamani, P. P. Widagdo y A. Irsyad, "Rancang Bangun Sistem Informasi Website Pergudangan Toko Mitra Mandiri Mebel Samarinda Berbasis Framework Laravel Dan Vue.Js Dengan Metode Waterfall," *Sains, Aplikasi, Komputasi dan Teknologi Informasi*, 2024. doi: 10.30872/jsakti.v5i1.13517.

[67] Docker, *What is Docker?* Nov. de 2025. dirección: https://docs.docker.com/get-started/docker-overview/.

[68] S. I. Shamim, J. A. Gibson, P. J. Morrison y A. A. U. Rahman, *Benefits, Challenges, and Research Topics: A Multi-vocal Literature Review of Kubernetes*, 2022. arXiv: 2211.07032 [cs.SE]. dirección: https://arxiv.org/abs/2211.07032.

[69] B. R. Cherukuri, "Microservices and containerization: Accelerating web development cycles," *World Journal of Advanced Research and Reviews*, vol. 6, págs. 283-296, 1 abr. de 2020. doi: 10.30574/wjarr.2020.6.1.0087.

[70] D. Patel et al., *Cloud Platforms for Developing Generative AI Solutions: A Scoping Review of Tools and Services*, 2024. arXiv: 2412.06044 [cs.DC]. dirección: https://arxiv.org/abs/2412.06044.

[71] Z. Xie, H. Kang, Y. Sheng, T. Krishna, K. Fatahalian y C. Kozyrakis, *AI Metropolis: Scaling Large Language Model-based Multi-Agent Simulation with Out-of-order Execution*, 2024. arXiv: 2411.03519 [cs.DC]. dirección: https://arxiv.org/abs/2411.03519.

[72] D. Meister, F. Dürr y F. Allgöwer, *Shared Network Effects in Time- versus Event-Triggered Consensus of a Single-Integrator Multi-Agent System*, 2023. arXiv: 2211.08156 [eess.SY]. dirección: https://arxiv.org/abs/2211.08156.

[73] A. Bemani y N. Björsell, "Distributed Event Triggering Algorithm for Multi-Agent System over a Packet Dropping Network," *Sensors*, vol. 21, n.º 14, 2021, issn: 1424-8220. doi: 10.3390/s21144835.

[74] K. Zhu et al., *MultiAgentBench: Evaluating the Collaboration and Competition of LLM agents*, 2025. arXiv: 2503.01935 [cs.MA]. dirección: https://arxiv.org/abs/2503.01935.

[75] X. Zhang, J. Yu y Z. Zhong, *Learning Efficient Communication Protocols for Multi-Agent Reinforcement Learning*, 2025. arXiv: 2511.09171 [cs.MA]. dirección: https://arxiv.org/abs/2511.09171.

[76] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger e Y. Artzi, *BERTScore: Evaluating Text Generation with BERT*, 2020. arXiv: 1904.09675 [cs.CL]. dirección: https://arxiv.org/abs/1904.09675.

[77] A. B. Sai, A. K. Mohankumar y M. M. Khapra, *A Survey of Evaluation Metrics Used for NLG Systems*, 2020. arXiv: 2008.12009 [cs.CL]. dirección: https://arxiv.org/abs/2008.12009.

[78] W. Zhao, M. Peyrard, F. Liu, Y. Gao, C. M. Meyer y S. Eger, *MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance*, K. Inui, J. Jiang, V. Ng y X. Wan, eds., Hong Kong, China, nov. de 2019. doi: 10.18653/v1/D19-1053.

[79] E. Clark, A. Celikyilmaz y N. A. Smith, *Sentence Mover's Similarity: Automatic Evaluation for Multi-Sentence Texts*, A. Korhonen, D. Traum y L. Màrquez, eds., Florence, Italy, jul. de 2019. doi: 10.18653/v1/P19-1264.

[80] M. Zhong et al., *Towards a Unified Multi-Dimensional Evaluator for Text Generation*, 2022. arXiv: 2210.07197 [cs.CL]. dirección: https://arxiv.org/abs/2210.07197.

[81] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu y C. Zhu, *G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment*, 2023. arXiv: 2303.16634 [cs.CL]. dirección: https://arxiv.org/abs/2303.16634.

[82] S. Es, J. James, L. Espinosa-Anke y S. Schockaert, *Ragas: Automated Evaluation of Retrieval Augmented Generation*, 2025. arXiv: 2309.15217 [cs.CL]. dirección: https://arxiv.org/abs/2309.15217.

[83] S. Dähling, L. Razik y A. Monti, "Enabling scalable and fault-tolerant multi-agent systems by utilizing cloud-native computing," *Autonomous Agents and Multi-Agent Systems*, vol. 35, 1 abr. de 2021, issn: 15737454. doi: 10.1007/s10458-020-09489-0.

[84] W. Jia y M. Ji, "Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control with Spatio-Temporal Attention Mechanism," *Applied Sciences*, vol. 15, n.º 15, 2025, issn: 2076-3417. doi: 10.3390/app15158605.

[85] L. Lomelí, *Metodología Scrum: Roles, Procesos y Artefactos*, mayo de 2023. dirección: https://innevo.com/blog/metodologia-scrum.

[86] pypi, *Gradio*, feb. de 2026. dirección: https://pypi.org/project/gradio/.

[87] OpenRouter, *LLM Rankings*. dirección: https://openrouter.ai/rankings?category=legal#categories.

ANEXOS

Tabla XXIV.
PUNTUACIONES G-EVAL POR MODELO (VALORES ORIGINALES)

Contract	Model	Lenovo				Acer			
		F	AR	CP	CR	F	AR	CP	CR
Adobe	Grok 4.1 Fast	1	0	1	1	1	0	1	1
	Gemini 2.5 Flash Lite	0,943	1	1	1	0,986	0	1	1
	Qwen3 VL 30B A3B Thinking	0,919	1	1	1	1	1	1	1
Anthropic	Grok 4.1 Fast	1	0	1	1	1	0	1	1
	Gemini 2.5 Flash Lite	0,963	0	1	1	1	0	1	1
	Qwen3 VL 30B A3B Thinking	1	0	1	1	0,974	0	1	1
Atlassian	Grok 4.1 Fast	1	0	1	1	0,929	0	1	1
	Gemini 2.5 Flash Lite	1	0	1	1	1	0	1	1
	Qwen3 VL 30B A3B Thinking	0,889	0	1	1	0,944	0	1	1
BlazeMeter	Grok 4.1 Fast	0,974	0	0,43	1	0,867	0	0,096	1
	Gemini 2.5 Flash Lite	0,8	0	0,093	1	1	0	0,098	1
	Qwen3 VL 30B A3B Thinking	1	0	0	1	0,929	0	0,097	1
Canva	Grok 4.1 Fast	1	0	1	1	1	0	1	1

Continuación de la Tabla XXIV

Contract	Model	Lenovo				Acer			
		F	AR	CP	CR	F	AR	CP	CR
	Gemini 2.5 Flash Lite	0,692	0	1	1	1	0	1	1
	Qwen3 VL 30B A3B Thinking	1	0	1	1	0,964	1	1	1
Figma	Grok 4.1 Fast	1	0	1	1	0,917	1	1	1
	Gemini 2.5 Flash Lite	1	1	1	1	1	1	1	1
	Qwen3 VL 30B A3B Thinking	1	0	1	1	0,974	0	1	1
GitHub	Grok 4.1 Fast	0,958	0	1	1	1	0	1	1
	Gemini 2.5 Flash Lite	0,875	0	1	1	0,871	0	1	1
	Qwen3 VL 30B A3B Thinking	0,864	0	1	1	0,952	0	1	1
HubSpot	Grok 4.1 Fast	1	0	1	1	1	0	1	1
	Gemini 2.5 Flash Lite	1	1	1	1	1	0	1	1
	Qwen3 VL 30B A3B Thinking	1	0	1	1	0,933	1	1	1
OpenAI	Grok 4.1 Fast	0,895	0	1	1	0,946	0	1	1
	Gemini 2.5 Flash Lite	0,875	1	1	1	1	0	1	1
	Qwen3 VL 30B A3B Thinking	0,852	0	1	1	0,957	0	1	1

Continuación de la Tabla XXIV

Contract	Model	Lenovo				Acer			
		F	AR	CP	CR	F	AR	CP	CR
Oracle	Grok 4.1 Fast	1	0	1	1	0,971	0	1	1
	Gemini 2.5 Flash Lite	1	0	1	1	0,929	0	1	1
	Qwen3 VL 30B A3B Thinking	0,955	1	1	1	1	1	1	1

Tabla XXV.
EVALUACIÓN HUMANA POR MODELO (VALORES ORIGINALES)

Contract	Model	Lenovo				Acer			
		F	AR	CP	CR	F	AR	CP	CR
Adobe	Grok 4.1 Fast	0,75	0,5	1	1	1	0,75	1	1
	Gemini 2.5 Flash Lite	0,50	0,25	0,75	0,75	0,75	0,5	0,75	0,75
	Qwen3 VL 30B A3B Thinking	0,75	0,75	0,75	0,75	1	1	1	1
Anthropic	Grok 4.1 Fast	1	0,5	0,75	1	0,75	0,75	0,75	0,75
	Gemini 2.5 Flash Lite	0	0	0,25	0,5	1	0,75	0,5	0,75
	Qwen3 VL 30B A3B Thinking	1	0,25	0,75	0,75	0,75	0,5	0,5	0,5
Atlassian	Grok 4.1 Fast	0,5	0	0,5	0,25	0,75	0,5	1	1
	Gemini 2.5 Flash Lite	1	0,75	0,75	0,25	0,25	0	0	0

Continuación de la Tabla XXV

Contract	Model	Lenovo				Acer			
		F	**AR**	**CP**	**CR**	**F**	**AR**	**CP**	**CR**
	Qwen3 VL 30B A3B Thinking	0,25	0,25	0	0,5	0,75	0,5	0,5	0,75
BlazeMeter	Grok 4.1 Fast	0,75	0,75	0,75	0,75	0,5	0,25	0,25	0,75
	Gemini 2.5 Flash Lite	0,25	0,5	0,5	0,5	0,5	0,5	0,5	0,5
	Qwen3 VL 30B A3B Thinking	0,75	0,75	1	1	0,25	0,5	1	0,5
Canva	Grok 4.1 Fast	1	1	1	1	1	1	1	1
	Gemini 2.5 Flash Lite	1	0,25	1	1	1	1	1	1
	Qwen3 VL 30B A3B Thinking	1	1	0,75	1	1	1	1	1
Figma	Grok 4.1 Fast	0,75	0,5	0,75	0,75	1	0,75	1	0,75
	Gemini 2.5 Flash Lite	1	1	1	1	0,75	0,5	0,25	0,25
	Qwen3 VL 30B A3B Thinking	0,75	0,75	0,75	0,75	1	0,75	0,75	0,75
GitHub	Grok 4.1 Fast	0,75	0,5	0,75	0,5	0,75	0,5	0,75	0,75
	Gemini 2.5 Flash Lite	0,25	0	0	0	0,75	0,5	0,75	0,75
	Qwen3 VL 30B A3B Thinking	0,5	0,5	0,75	1	0,5	0,25	0,25	0,25
HubSpot	Grok 4.1 Fast	0,5	0,25	0,5	0,75	0,5	0,25	0,75	0,75

Continuación de la Tabla XXV

Contract	Model	Lenovo				Acer			
		F	**AR**	**CP**	**CR**	**F**	**AR**	**CP**	**CR**
	Gemini 2.5 Flash Lite	0	0	0	0	0	0	0	0
	Qwen3 VL 30B A3B Thinking	0,25	0	0	0	0,25	0	0,25	0,25
OpenAI	Grok 4.1 Fast	1	0,75	1	1	1	0,75	1	1
	Gemini 2.5 Flash Lite	0,75	0,5	1	0,5	0,5	0,5	0,5	0,5
	Qwen3 VL 30B A3B Thinking	0,75	0,5	1	1	1	0,5	1	1
Oracle	Grok 4.1 Fast	1	0,5	0,75	1	0,75	0,5	0,75	0,75
	Gemini 2.5 Flash Lite	0,5	0,75	0,75	0,75	0,75	1	0,75	0,75
	Qwen3 VL 30B A3B Thinking	0,5	0,25	0,75	0,75	0,75	0,75	0,5	0,5

Tabla XXVI.

COMPARATIVA G-EVAL VS. EVALUACIÓN HUMANA POR MODELO Y MÉTRICA (VALORES ORIGINALES)

Contrato	Modelo	G-Eval								Evaluación Humana							
		Lenovo				Acer				Lenovo				Acer			
		F	AR	CP	CR	F	AR	CP	CR	F	AR	CP	CR	F	AR	CP	CR
Adobe	Grok 4.1 Fast	1	0	1	1	1	0	1	1	0,75	0,5	1	1	1	0,75	1	1
	Gemini 2.5 Flash Lite	0,943	1	1	1	0,986	0	1	1	0,50	0,25	0,75	0,75	0,75	0,5	0,75	0,75
	Qwen3 VL 30B A3B Thinking	0,919	1	1	1	1	1	1	1	0,75	0,75	0,75	0,75	1	1	1	1
Anthropic	Grok 4.1 Fast	1	0	1	1	1	0	1	1	1	0,5	0,75	1	0,75	0,75	0,75	0,75
	Gemini 2.5 Flash Lite	0,963	0	1	1	1	0	1	1	0	0	0,25	0,5	1	0,75	0,5	0,75
	Qwen3 VL 30B A3B Thinking	1	0	1	1	0,974	0	1	1	1	0,25	0,75	0,75	0,75	0,5	0,5	0,5
Atlassian	Grok 4.1 Fast	1	0	1	1	0,929	0	1	1	0,5	0	0,5	0,25	0,75	0,5	1	1

Continuación de la Tabla XXVI

Contrato	Modelo	G-Eval								Evaluación Humana							
		Lenovo				Acer				Lenovo				Acer			
		F	AR	CP	CR	F	AR	CP	CR	F	AR	CP	CR	F	AR	CP	CR
	Gemini 2.5 Flash Lite	1	0	1	1	1	0	1	1	1	0,75	0,75	0,25	0,25	0	0	0
	Qwen3 VL 30B A3B Thinking	0,889	0	1	1	0,944	0	1	1	0,25	0,25	0	0,5	0,75	0,5	0,5	0,75
BlazeMeter	Grok 4.1 Fast	0,974	0	0,43	1	0,867	0	0,096	1	0,75	0,75	0,75	0,75	0,5	0,25	0,25	0,75
	Gemini 2.5 Flash Lite	0,8	0	0,093	1	1	0	0,098	1	0,25	0,5	0,5	0,5	0,5	0,5	0,5	0,5
	Qwen3 VL 30B A3B Thinking	1	0	0	1	0,929	0	0,097	1	0,75	0,75	1	1	0,25	0,5	1	0,5
Canva	Grok 4.1 Fast	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	Gemini 2.5 Flash Lite	0,692	0	1	1	1	0	1	1	1	0,25	1	1	1	1	1	1

Continuación de la Tabla XXVI

Contrato	Modelo	G-Eval								Evaluación Humana							
		Lenovo				Acer				Lenovo				Acer			
		F	AR	CP	CR	F	AR	CP	CR	F	AR	CP	CR	F	AR	CP	CR
	Qwen3 VL 30B A3B Thinking	1	0	1	1	0,964	1	1	1	1	1	0,75	1	1	1	1	1
Figma	Grok 4.1 Fast	1	0	1	1	0,917	1	1	1	0,75	0,5	0,75	0,75	1	0,75	1	0,75
	Gemini 2.5 Flash Lite	1	1	1	1	1	1	1	1	1	1	1	1	0,75	0,5	0,25	0,25
	Qwen3 VL 30B A3B Thinking	1	0	1	1	0,974	0	1	1	0,75	0,75	0,75	0,75	1	0,75	0,75	0,75
GitHub	Grok 4.1 Fast	0,958	0	1	1	1	0	1	1	0,75	0,5	0,75	0,5	0,75	0,5	0,75	0,75
	Gemini 2.5 Flash Lite	0,875	0	1	1	0,871	0	1	1	0,25	0	0	0	0,75	0,5	0,75	0,75
	Qwen3 VL 30B A3B Thinking	0,864	0	1	1	0,952	0	1	1	0,5	0,5	0,75	1	0,5	0,25	0,25	0,25
HubSpot	Grok 4.1 Fast	1	0	1	1	1	0	1	1	0,5	0,25	0,5	0,75	0,5	0,25	0,75	0,75

Continuación de la Tabla XXVI

Contrato	Modelo	G-Eval								Evaluación Humana							
		Lenovo				Acer				Lenovo				Acer			
		F	AR	CP	CR	F	AR	CP	CR	F	AR	CP	CR	F	AR	CP	CR
	Gemini 2.5 Flash Lite	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
	Qwen3 VL 30B A3B Thinking	1	0	1	1	0,933	1	1	1	0,25	0	0	0	0,25	0	0,25	0,25
OpenAI	Grok 4.1 Fast	0,895	0	1	1	0,946	0	1	1	1	0,75	1	1	1	0,75	1	1
	Gemini 2.5 Flash Lite	0,875	1	1	1	1	0	1	1	0,75	0,5	1	0,5	0,5	0,5	0,5	0,5
	Qwen3 VL 30B A3B Thinking	0,852	0	1	1	0,957	0	1	1	0,75	0,5	1	1	1	0,5	1	1
Oracle	Grok 4.1 Fast	1	0	1	1	0,971	0	1	1	1	0,5	0,75	1	0,75	0,5	0,75	0,75
	Gemini 2.5 Flash Lite	1	0	1	1	0,929	0	1	1	0,5	0,75	0,75	0,75	0,75	1	0,75	0,75

Continuación de la Tabla XXVI

Contrato	Modelo	G-Eval								Evaluación Humana							
		Lenovo				Acer				Lenovo				Acer			
		F	AR	CP	CR	F	AR	CP	CR	F	AR	CP	CR	F	AR	CP	CR
	Qwen3 VL 30B A3B Thinking	0,955	1	1	1	1	1	1	1	0,5	0,25	0,75	0,75	0,75	0,75	0,5	0,5