Exploración de agentes de IA con LLMs locales: Evaluación en procesos de recuperación, transcripción y renombramiento de información

Edgar Eduardo Alvarez Rengifo

Proyecto de grado entregado para obtener el título de
Magister en Ingeniería de Software

Dirigida por
Ph.D. Luisa Rincón

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Maestría en Ingeniería de Software
Santiago de Cali
10 de Julio de 2026

Santiago de Cali, 10 de Julio de 2026.

Señores
Pontificia Universidad Javeriana Cali.
Jorge Enrique Alvarez Patiño
Director Posgrados de Ingeniería.
Cali.

Cordial Saludo.

Por medio de la presente hago constar que en mi calidad de director de trabajo de grado he revisado el proyecto titulado "Exploración de agentes de IA con LLMs locales: Evaluación en procesos de recuperación, transcripción y renombramiento de información" realizado por el estudiante de Magister en Ingeniería de Software Edgar Eduardo Alvarez Rengifo (cod: 1151956769), el cual se encuentra terminado y considero que cumple con los requisitos para ser sustentado.
Atentamente,

Digitally signed by Luisa Rincón
DN: OU=Dpto Elect y Ciencias Comput - Facultad Ing y Ciencias, O=Pontificia Universidad Javeriana Cali, CN=Luisa Rincón, E=lfrincon@javerianacali.edu.co
Reason: I agree to specified parts of this document
Location:
Date: 2026.07.10 12:20:49-05'00'
Foxit PDF Reader Version: 2025.2.0

Ph.D. Luisa Rincón
CC. 1058817701

Santiago de Cali, 10 de Julio de 2026.

Señores
Pontificia Universidad Javeriana Cali
Jorge Enrique Alvarez Patiño
Director Posgrados de Ingeniería.
Cali.

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado "Exploración de agentes de IA con LLMs locales: Evaluación en procesos de recuperación, transcripción y renombramiento de información" con el fin de cumplir con los requisitos exigidos por la Universidad y para que sea sometido a revisión del jurado y cumpla su aprobación, para conseguir posteriormente el título de Magister en Ingeniería de Software.

Atentamente,

Edgar Alvarez

Edgar Eduardo Alvarez Rengifo
Código: 1151956769

Ficha Resumen
Trabajo de Grado Maestría en Ingeniería de Software

TíTULO: Exploración de agentes de IA con LLMs locales: Evaluación en procesos de recuperación, transcripción y renombramiento de información

1. Énfasis: Ingeniería de Software
2. Área de trabajo: Inteligencia Artificial
3. Tipo de proyecto (Aplicado, Innovación, Investigación): : Aplicado
4. Estudiante: Edgar Eduardo Alvarez Rengifo
5. Correo electrónico: ealvarez45@javerianacali.edu.co
6. Dirección y teléfono: calle 59 norte 3 b n 24, 3054300233
7. Director: PhD. Luisa Rincón
8. Vinculación del director: (Planta/Cátedra/Externo): Docente de planta - Pontificia Universidad Javeriana Cali
9. Correo electrónico del director: lfrincon@javerianacali.edu.co
10. Co-Director (Si aplica):
11. Grupo o empresa que lo avala (Si aplica):
12. Otros grupos o empresas:
13. Palabras clave(al menos 5): LLMs, Automation, artificial intelligence agents, academic-administrative processes, higher education, Scrum, digital transformation.
14. ODS que aplica al proyecto (Agenda 2030):
15. Fecha de inicio: 1 de Julio de 2025
16. Resumen: Este trabajo de grado evaluó la viabilidad de implementar agentes de inteligencia artificial basados en modelos de lenguaje grandes (LLMs) ejecutados localmente en entornos institucionales con restricciones de privacidad. Se analizaron los requerimientos técnicos, los costos de implementación y las implicaciones de seguridad asociadas al uso de infraestructura on-premise frente a soluciones en la nube. Posteriormente, se diseñaron e implementaron tres prototipos funcionales orientados a procesos académico-administrativos: recuperación aumentada por generación (RAG), transcripción automática y renombramiento semántico de documentos. La evaluación se realizó mediante pruebas funcionales, pruebas de carga

y sesiones con usuarios, considerando métricas de precisión, eficiencia operativa, facilidad de implementación y adecuación al contexto institucional. Los resultados mostraron que la ejecución local es viable en escenarios controlados, con ventajas en el control de datos y costos estables, aunque con limitaciones en escalabilidad para tareas intensivas en cómputo. Finalmente, se propusieron lineamientos para la adopción institucional de agentes de IA.

Agradecimientos

Quiero expresar mi agradecimiento a todas las personas que, de una u otra manera, hicieron posible el desarrollo de este trabajo y me acompañaron durante el proceso de la maestría. La elaboración de esta investigación representó un reto académico y personal que no habría sido posible afrontar sin el apoyo, la orientación y la paciencia de quienes estuvieron presentes en distintas etapas del camino.

A mi familia, por su apoyo constante, comprensión y confianza durante estos años. Gracias por acompañarme en los momentos de mayor carga académica, por recordarme la importancia de continuar aun cuando el proceso parecía más difícil de lo esperado. Su respaldo fue fundamental para llegar hasta este punto.

A mi directora de trabajo de grado, por su orientación, disposición y rigurosidad académica. Sus observaciones, preguntas y recomendaciones ayudaron a fortalecer este proyecto y me llevaron a profundizar aspectos técnicos y metodológicos que inicialmente no había considerado. Más allá de las correcciones, agradezco especialmente el acompañamiento durante un proyecto que evolucionó constantemente y que exigió replantear ideas en varias ocasiones.

A los profesores de la Maestría en Ingeniería de Software de la Pontificia Universidad Javeriana Cali, por los conocimientos compartidos y las discusiones académicas que contribuyeron a ampliar mi perspectiva sobre la ingeniería de software, la inteligencia artificial y la toma de decisiones tecnológicas en contextos reales.

A las personas y colegas que, desde conversaciones técnicas, recomendaciones o intercambio de experiencias, aportaron ideas para el desarrollo de los prototipos y la comprensión de tecnologías relacionadas con modelos de lenguaje, automatización y despliegues locales.

Finalmente, agradezco a la Pontificia Universidad Javeriana Cali por brindar el espacio académico para desarrollar esta investigación y por promover escenarios donde es posible explorar tecnologías emergentes desde una mirada crítica, aplicada y orientada a resolver problemas reales.

Mirando el proceso en retrospectiva, este trabajo no solo representó un ejercicio de investigación sobre inteligencia artificial y despliegues locales, sino también una oportunidad para aprender a enfrentar problemas abiertos, cuestionar supuestos iniciales y comprender que detrás de cada decisión técnica existen implicaciones humanas, organizacionales y éticas. Todo ello hizo de esta experiencia una etapa de crecimiento personal y profesional que recordaré con gratitud.

0.1. Resumen

Este trabajo de grado evaluó la viabilidad de implementar agentes de inteligencia artificial basados en modelos de lenguaje grandes (LLMs) ejecutados localmente en entornos institucionales con restricciones de privacidad. Se analizaron los requerimientos técnicos, los costos de implementación y las implicaciones de seguridad asociadas al uso de infraestructura on-premise frente a soluciones en la nube. Posteriormente, se diseñaron e implementaron tres prototipos funcionales orientados a procesos académico-administrativos: recuperación aumentada por generación (RAG), transcripción automática y renombramiento semántico de documentos. La evaluación se realizó mediante pruebas funcionales, pruebas de carga y sesiones con usuarios, considerando métricas de precisión, eficiencia operativa, facilidad de implementación y adecuación al contexto institucional. Los resultados mostraron que la ejecución local es viable en escenarios controlados, con ventajas en el control de datos y costos estables, aunque con limitaciones en escalabilidad para tareas intensivas en cómputo. Finalmente, se propusieron lineamientos para la adopción institucional de agentes de IA.

Palabras Clave: inteligencia artificial, modelos de lenguaje, LLMs locales, RAG, transcripción automática, agentes de IA, privacidad de datos, infraestructura on-premise

0.2. Abstract

This study evaluated the feasibility of implementing artificial intelligence agents based on large language models (LLMs) executed locally in institutional environments with data privacy constraints. Technical requirements, implementation costs, and security implications were analyzed, comparing on-premise infrastructure with cloud-based solutions. Subsequently, three functional prototypes were designed and implemented for academic-administrative processes: retrieval-augmented generation (RAG), automatic transcription, and semantic file renaming. The evaluation was conducted through functional testing, load testing, and user sessions, considering metrics such as accuracy, operational efficiency, ease of implementation, and alignment with institutional needs. The results showed that local deployment is feasible in controlled scenarios, offering advantages in data control and cost stability, although with scalability limitations for computationally intensive tasks. Finally, guidelines for the institutional adoption of AI agents were proposed.

Keywords: artificial intelligence, large language models, local LLMs, retrieval-augmented generation, automatic transcription, AI agents, data privacy, on-premise infrastructure

Índice general

Agradecimientos **7**
0.1. Resumen . . . 8
0.2. Abstract . . . 8

1. Introducción **1**
1.1. Definición del problema de investigación . . . 1
1.2. Planteamiento del problema . . . 1
1.3. Sistematización . . . 2
1.4. Objetivos . . . 2
1.4.1. Objetivo General . . . 2
1.4.2. Objetivos Específicos . . . 3
1.5. Justificación . . . 4
1.6. Delimitaciones y Alcances . . . 4
1.7. Resultados Obtenidos . . . 5

2. Marco de referencia **7**
2.1. Bases teóricas . . . 7
2.2. Estado del arte . . . 12

3. Requerimientos para la adopción de LLMs locales en entornos institucionales **17**
3.1. Identificación de requerimientos técnicos para la ejecución local de LLMs . . . 17
3.1.1. Recolección de información . . . 18
3.1.2. Matriz comparativa de requerimientos técnicos (2025–2026) . . . 19
3.1.3. Análisis de los requerimientos identificados . . . 19
3.1.4. Rangos típicos de infraestructura por tamaño de modelo . . . 20
3.1.5. Frameworks y ecosistema de despliegue . . . 21
3.1.6. Conclusiones técnicas . . . 22
3.1.7. Recomendaciones . . . 23
3.2. Análisis de costos de implementación y mantenimiento de agentes de IA con LLMs locales . . . 23
3.2.1. Estructura de costos evaluados . . . 24
3.2.2. Requerimientos de infraestructura local . . . 24
3.2.3. Costos estimados de infraestructura local . . . 25
3.2.4. Costos basados en la nube . . . 27
3.2.5. Proyección consolidada a uno, dos y tres años . . . 28
3.3. Implicaciones de privacidad en el uso de LLMs en infraestructura propia . . . 30
3.3.1. Contexto normativo y ético . . . 30

3.3.2. Riesgos de privacidad en el uso de LLMs locales . . . 31
3.3.3. Análisis de vulnerabilidades y evaluación de impacto . . . 32
3.3.4. Recomendaciones para la mitigación de riesgos . . . 33
3.3.5. Síntesis de riesgos y consideraciones finales . . . 36
3.3.6. Análisis integrado de hallazgos y conclusiones del capítulo . . . 37
3.4. Síntesis . . . 38

4. Diseño e implementación de prototipos 39
4.1. Diseño e implementación de prototipos funcionales de agentes de IA con LLMs locales 39
4.1.1. Contexto institucional y supuestos de implementación . . . 39
4.2. Selección y justificación de los casos de uso . . . 40
4.3. Arquitectura general . . . 42
4.3.1. Sobre Ollama y por qué se eligió . . . 43
4.4. Enfoque de diseño e implementación . . . 44
4.4.1. Metodología de desarrollo . . . 45
4.5. Caso de uso 1: Recuperación aumentada por generación (RAG) . . . 46
4.5.1. Necesidad que soporta . . . 46
4.5.2. Interacción del usuario . . . 46
4.5.3. Responsabilidades del área de TI . . . 46
4.5.4. Despliegue de la solución . . . 46
4.5.5. Descripción del prototipo . . . 47
4.6. Caso de uso 2: Transcripción automática . . . 48
4.6.1. Necesidad que soporta . . . 48
4.6.2. Interacción del usuario . . . 48
4.6.3. Responsabilidades del área de TI . . . 48
4.6.4. Despliegue de la solución . . . 49
4.6.5. Descripción del prototipo . . . 49
4.7. Caso de uso 3: Renombramiento semántico de documentos . . . 50
4.7.1. Necesidad que soporta . . . 50
4.7.2. Interacción del usuario . . . 50
4.7.3. Responsabilidades del área de TI . . . 51
4.7.4. Despliegue de la solución . . . 51
4.7.5. Descripción del prototipo . . . 52
4.8. Herramientas, entregables y código fuente . . . 53
4.8.1. Herramientas utilizadas . . . 53
4.8.2. Entregables . . . 54
4.8.3. Capturas de las soluciones en uso . . . 55
4.9. Síntesis . . . 57

5. Evaluación **59**
5.1. Evaluación del desempeño de los prototipos de agentes de IA 59
5.1.1. Metodología de evaluación 59
5.1.2. Criterios de aceptación 59
5.1.3. Entorno de pruebas 60
5.1.4. Diseño de las pruebas no funcionales 61
5.1.5. Resultados por caso de uso 62
5.1.6. Evaluación de percepción con la Dirección de Tecnologías de Información . . 69
5.1.7. Evaluación integrada del desempeño 72
5.1.8. Interpretación de resultados 73

6. Conclusiones **75**
6.1. Conclusiones 75
6.1.1. Conclusiones por objetivo 75
6.1.2. Lecciones aprendidas 76
6.1.3. Conclusión general 77
6.2. Discusión y reflexión final sobre el proceso 77
6.2.1. Si tuviera que decidir hoy: por qué opción me inclinaría 78
6.2.2. Lo más fácil y lo más difícil de configurar 78
6.2.3. Lo que aprendí durante el proceso 79
6.2.4. Trabajo futuro 80

Bibliografía **81**

Índice de figuras

2.1. Modelo de procesos (Pontificia Universidad Javeriana Cali, 2021). 7
2.2. Ciclo de formación al estudiante (Pontificia Universidad Javeriana Cali, 2021). . . . 8
2.3. Diferencias entre las interacciones con modelos de lenguaje grandes (LLMs) mediante acción directa en comparación con el uso de agentes proxy, agentes funcionales y agentes autónomos (Lanham, 2025). 11

3.1. Comparación de costos totales estimados (USD) entre ejecución local y en la nube a 1, 2 y 3 años 29

4.1. Diagrama de despliegue del agente RAG. Los tres nodos pueden corresponder a una sola máquina (despliegue de prototipo, como en este trabajo, donde los tres contenedores corren en el mismo equipo de pruebas descrito en la Tabla 5.1 del Capítulo 5) o a tres máquinas distintas en una arquitectura productiva, comunicadas por la red institucional. La elección depende del volumen de usuarios concurrentes esperado: para los rangos discutidos en §3.1.4, una sola máquina con 16–24 GB de VRAM es suficiente. 47
4.2. Agente RAG — Recuperación Aumentada por Generación 48
4.3. Diagrama de despliegue del agente de transcripción automática 49
4.4. Arquitectura del prototipo de transcripción de archivos multimedia con n8n, servicio de transcripción y Ollama. 50
4.5. Diagrama de despliegue del agente de renombramiento semántico. En el prototipo, los nodos del agente y el motor Ollama corren en la misma máquina; en un despliegue productivo pueden separarse para que un único servidor de inferencia (Ollama) atienda a varios servicios de la institución, incluyendo este agente y el RAG del caso de uso 1. 52
4.6. Arquitectura del prototipo multiagente de renombramiento automatizado con AutoGen, Ollama y directorios locales de entrada y salida. 53
4.7. Captura de Open WebUI: el usuario formula una pregunta en lenguaje natural y recibe la respuesta del agente RAG con la cita al documento institucional consultado. 56
4.8. Captura del editor de flujos de n8n: el flujo de transcripción muestra los nodos de carga del archivo, llamada al servicio Whisper y postprocesamiento con Ollama. Cada nodo es configurable desde la propia interfaz. 56
4.9. Captura del agente de renombramiento: directorios de entrada y salida con archivos antes y después del procesamiento. El nombre asignado por el agente sigue el formato definido en las reglas de nomenclatura. 57

5.1. Arquitectura escalable para el agente RAG mediante balanceo de carga y múltiples instancias de inferencia 65
5.2. Arquitectura escalable del agente de transcripción utilizando cola de trabajos y múltiples workers 67

Índice de tablas

3.1. Familias de LLMs evaluadas y sus organizaciones desarrolladoras 18
3.2. Matriz comparativa de requerimientos técnicos para LLMs locales (2025–2026). El consumo de GPU corresponde al rango típico durante inferencia continua según las TDP publicadas por el fabricante de la GPU asociada a cada categoría de hardware. 19
3.3. Requerimientos técnicos para infraestructura local 25
3.4. Costos estimados para infraestructura local (*on-premise*) 26
3.5. Tarifas de inferencia por proveedor (USD por millón de tokens, primer trimestre de 2026) 28
3.6. Costos mensuales y anuales en la nube por proveedor (USD) 28
3.7. Comparación de costos totales acumulados a 1, 2 y 3 años (USD) 29
3.8. Riesgos potenciales de privacidad en LLMs locales 32
3.9. Medidas recomendadas de mitigación de riesgos basadas en marcos normativos y estándares 34
3.10. Matriz consolidada de riesgos de privacidad y medidas de mitigación 37

4.1. Modelos relevantes del catálogo de Ollama y su idoneidad por tarea 44
4.2. Herramientas y tecnologías utilizadas en el desarrollo de los prototipos 54

5.1. Especificaciones del entorno de pruebas 61
5.2. Resultados de la evaluación funcional del agente RAG (30 preguntas) 63
5.3. Resultados de la evaluación funcional del agente de renombramiento (50 documentos) 68
5.4. Estructura de las sesiones de evaluación con la DTI 70

Capítulo 1

Introducción

1.1. Definición del problema de investigación

1.2. Planteamiento del problema

El uso de modelos de lenguaje grandes (LLMs, por sus siglas en inglés) ha transformado significativamente la manera en que se abordan tareas complejas de procesamiento de lenguaje natural, automatización de flujos de trabajo y generación de conocimiento (Zhang et al., 2024). Estas tecnologías permiten desarrollar agentes de inteligencia artificial capaces de asistir en tareas como recuperación de información, clasificación semántica, generación de texto y análisis contextual, lo cual ha ampliado su adopción en sectores como la salud, la educación y la industria.

No obstante, la adopción de LLMs en instituciones académicas y organizaciones que manejan datos sensibles enfrenta barreras importantes, especialmente en lo relacionado con la privacidad, la seguridad de la información y la dependencia tecnológica. Diversos estudios han documentado riesgos asociados al uso de modelos de IA comerciales como ChatGPT (OpenAI), Gemini (Google) o Claude (Anthropic), entre ellos: la retención de datos ingresados por el usuario, su uso para entrenar modelos sin consentimiento explícito y la exposición involuntaria de información sensible en sistemas en la nube (Li, 2023); (Das et al., 2024).

En contextos como el sector salud, organizaciones gubernamentales o entidades educativas, estas prácticas pueden violar regulaciones como el GDPR o HIPAA, comprometiendo la confidencialidad de los datos procesados. Por ejemplo, en 2023, el regulador de datos italiano suspendió temporalmente el uso de ChatGPT por preocupaciones sobre cómo gestionaba datos personales (ICT&Health, 2025). Asimismo, empresas como Samsung y JPMorgan Chase prohibieron el uso de IA generativa a sus empleados luego de filtraciones internas no intencionales al introducir información confidencial en herramientas como ChatGPT (Gal, 2023).

Un análisis de las políticas de privacidad de estas plataformas revela diferencias importantes. OpenAI, por ejemplo, permite utilizar los datos de las conversaciones para entrenar sus modelos por defecto, salvo que el usuario desactive manualmente esta opción o utilice un plan empresarial que garantice exclusión del entrenamiento (OpenAI, 2026). Google Gemini mantiene activa la recolección de entradas por defecto en usuarios individuales, aunque ofrece garantías contractuales en Google Workspace para clientes institucionales (Google, 2025). Anthropic, en contraste, adopta una postura más restrictiva: no utiliza los datos del usuario para entrenamiento, salvo que se otorgue consentimiento explícito (Anthropic, 2023).

Estas condiciones cuestionan la idoneidad de las soluciones comerciales en instituciones como la Pontificia Universidad Javeriana Cali, que manejan información académica, personal y administra-

tiva sensible. Es por ello que cada vez más organizaciones están explorando alternativas de código abierto que puedan ejecutarse de forma local (on-premise), brindando mayor control, auditabilidad y alineación con políticas institucionales y marcos normativos (Dennstädt et al., 2025).

La Pontificia Universidad Javeriana Cali cuenta con un modelo de procesos estructurado en tres macrociclos: estratégicos, focales y de apoyo. Dentro del ciclo focal de formación del estudiante se agrupan los procesos de gestión académico-administrativa, cuyo propósito es brindar soporte operativo para facilitar el desarrollo de las actividades formativas (Pontificia Universidad Javeriana Cali, 2021). A pesar de los avances en digitalización, aún persisten procesos que dependen de tareas manuales repetitivas, como la clasificación y actualización de registros, el envío de comunicaciones o la consolidación de información. Esta dependencia de la intervención manual genera desafíos como errores humanos, demoras en la atención, sobrecarga laboral del personal y una disminución en la calidad del servicio ofrecido a estudiantes y usuarios internos (Chugh et al., 2022).

Ante este escenario, se hace necesaria una evaluación práctica y contextualizada sobre la adopción de agentes de IA con LLMs locales. Este proyecto examinó la viabilidad desde las dimensiones técnica, económica y de privacidad de datos, mediante la implementación de prototipos aplicados a tres casos de uso institucionales: recuperación aumentada por generación (RAG), transcripción de archivos multimedia y renombramiento inteligente de documentos. Los resultados obtenidos ofrecen evidencia para futuras decisiones de automatización bajo entornos controlados y con respeto por la privacidad institucional.

1.3. Sistematización

- ¿Qué requerimientos técnicos (infraestructura de hardware, recursos computacionales, configuración) son necesarios para ejecutar LLMs de manera local en un entorno institucional?
- ¿Cuáles son los costos asociados a la implementación y mantenimiento de LLMs locales en comparación con soluciones en la nube?
- ¿Qué implicaciones de privacidad deben considerarse al adoptar agentes basados en LLMs en una institución educativa?
- ¿Qué tan eficaces son los agentes de IA con LLMs locales en tres casos de uso específicos: recuperación aumentada por generación (RAG), transcripción de archivos multimedia y renombramiento automático de archivos?

1.4. Objetivos

1.4.1. Objetivo General

Evaluar la viabilidad de implementar agentes de inteligencia artificial basados en modelos de lenguaje grandes (LLMs) de ejecución local para automatizar procesos académico-administrativos en entornos institucionales con restricciones de privacidad, a través de tres casos de uso en la Pontificia Universidad Javeriana Cali.

1.4.2. Objetivos Específicos

1. **Identificar** los requerimientos técnicos necesarios para la ejecución local de modelos de lenguaje grandes (LLMs) en entornos institucionales.

2. **Analizar** los costos de implementación y mantenimiento de agentes de IA con LLMs locales, en comparación con soluciones basadas en la nube.

3. **Examinar** las implicaciones de privacidad que conlleva el uso de LLMs en infraestructura propia, dentro del contexto universitario.

4. **Diseñar e implementar** prototipos funcionales de agentes de IA con LLMs locales aplicados a tres procesos académico-administrativos: recuperación aumentada por generación (RAG), transcripción de archivos multimedia y renombramiento automatizado de archivos.

5. **Evaluar** el desempeño de los prototipos en cuanto a eficiencia operativa, percepción de utilidad y alineación con las necesidades institucionales, proponiendo lineamientos para su posible adopción futura.

1.5. Justificación

La creciente adopción de modelos de lenguaje grandes (LLMs) ha abierto nuevas oportunidades para automatizar tareas complejas en diversos sectores. Sin embargo, muchas instituciones académicas, como la Pontificia Universidad Javeriana Cali, enfrentan restricciones en cuanto al uso de tecnologías en la nube debido a preocupaciones sobre privacidad, cumplimiento normativo y control sobre la información institucional (Das et al., 2024). En este contexto, surge la necesidad de explorar alternativas locales que permitan aprovechar los beneficios de la inteligencia artificial sin comprometer la seguridad ni la soberanía de los datos (Radas et al., 2024).

Este proyecto se justifica en primer lugar por su aporte a la toma de decisiones institucional, ya que permitió evaluar de forma sistemática la viabilidad técnica, económica y de seguridad de implementar agentes de IA con LLMs locales en el contexto de procesos académico-administrativos. Al centrar el análisis en tres casos de uso concretos (RAG, transcripción y renombramiento), se proporcionaron evidencias aplicables y realistas que pueden orientar futuras estrategias de automatización dentro de la universidad. Diversos estudios han documentado los beneficios potenciales de los LLMs en la mejora de servicios educativos, siempre que se gestionen adecuadamente los riesgos de privacidad y se adapten a contextos locales (Zhang et al., 2024).

Desde una perspectiva tecnológica, el proyecto contribuye a cerrar una brecha actual en la literatura y en la práctica, al documentar con rigor los retos y oportunidades de ejecutar modelos LLM en entornos locales. Esto incluye el análisis de requerimientos de infraestructura, desempeño y sostenibilidad, aspectos que han sido identificados como críticos en estudios recientes (Samsi et al., 2023). Además, se promueve el uso de tecnologías de código abierto, lo que favorece la autonomía tecnológica y reduce la dependencia de proveedores externos (Dennstädt et al., 2025).

Finalmente, desde el punto de vista académico, el proyecto representa una oportunidad para aplicar conocimientos avanzados en inteligencia artificial, ingeniería de software y gestión de tecnología, en un entorno real y con impacto directo en la mejora de los servicios educativos. Los resultados podrán servir como base para futuras investigaciones y desarrollos orientados a la transformación digital segura de instituciones de educación superior (Dorca Josa and Bleda-Bejar, 2024).

1.6. Delimitaciones y Alcances

Este proyecto se desarrolló en un periodo de cinco meses con una dedicación de 192 horas, y se centró en la evaluación de la viabilidad técnica, económica y de privacidad para implementar agentes de inteligencia artificial basados en modelos de lenguaje grandes (LLMs) de ejecución local, aplicados a procesos académico-administrativos en la Pontificia Universidad Javeriana Cali.

El proyecto no contempló el despliegue en producción de los agentes desarrollados, ni el acceso directo a datos sensibles institucionales. Todas las pruebas se realizaron en entornos simulados con datos previamente anonimizados. El desarrollo se limitó a tecnologías de código abierto y ejecutables en infraestructura local, sin depender de servicios comerciales en la nube.

El alcance de la evaluación estuvo definido por las condiciones en las que se desarrolló el pro-

yecto. Durante su ejecución, la Universidad estaba comenzando a explorar el uso de LLMs locales, por lo que se optó por trabajar con escenarios que pudieran implementarse y evaluarse con la infraestructura disponible. De igual manera, todas las pruebas se realizaron con documentos de acceso público, ya que el proceso requerido para autorizar el uso de información confidencial excedía el tiempo previsto para este trabajo. A esto se sumó el periodo de ejecución de aproximadamente cuatro meses y medio, lo que hizo necesario concentrar el esfuerzo en tres casos de uso y en un conjunto acotado de pruebas.

Los resultados obtenidos corresponden a las condiciones bajo las cuales fueron desarrollados y evaluados los prototipos. Por ello, las conclusiones de este trabajo se limitan al corpus utilizado, a la infraestructura disponible y a las configuraciones implementadas durante la evaluación.

1.7. Resultados Obtenidos

El desarrollo del proyecto permitió obtener resultados en cuatro frentes principales relacionados con la viabilidad de implementar agentes de inteligencia artificial basados en LLMs locales dentro de un contexto institucional.

En primer lugar, se identificaron los requerimientos mínimos de infraestructura necesarios para ejecutar modelos de lenguaje grandes en entornos institucionales, considerando variables como memoria RAM, VRAM, capacidad de almacenamiento, consumo energético y complejidad de despliegue. Este análisis permitió establecer rangos de configuración para distintos tamaños de modelo y escenarios de uso.

En segundo lugar, se realizó un análisis comparativo de costos entre el despliegue local (*on-premise*) y el uso de servicios basados en la nube, incluyendo estimaciones de inversión inicial, costos operativos y gastos de mantenimiento. La comparación permitió identificar en qué condiciones una infraestructura local resulta más conveniente para una institución educativa y cuándo un enfoque híbrido puede representar una alternativa más adecuada.

En tercer lugar, se diseñaron e implementaron tres prototipos funcionales de agentes de IA ejecutados localmente y evaluados en un entorno controlado, aplicados a los siguientes casos de uso:

- Recuperación aumentada por generación (*Retrieval-Augmented Generation*, RAG), orientada a consultas sobre documentación institucional.
- Transcripción automática de archivos de audio y video mediante modelos de reconocimiento de voz.
- Renombramiento semántico de documentos administrativos a partir de su contenido.

Finalmente, se obtuvo una evaluación experimental del comportamiento de los prototipos mediante pruebas funcionales, pruebas de carga y sesiones de validación con usuarios institucionales, lo que permitió identificar fortalezas, limitaciones operativas y condiciones de adopción para escenarios reales de aplicación.

Capítulo 2

Marco de referencia

2.1. Bases teóricas

Modelo de Procesos de la Pontificia Universidad Javeriana Cali

Un modelo de procesos empresariales muestra cómo se espera que se realicen las tareas, quién es responsable de ellas y cómo un proceso contribuye al logro de un objetivo empresarial. Esto desempeña un papel importante en el aumento de la confianza y la rendición de cuentas en toda la organización (Kay, 2020).

La Pontificia Universidad Javeriana Cali cuenta con un modelo de procesos que se divide en tres macro ciclos (estratégicos, focales y de apoyo) como se puede observar en la Figura 2.1.

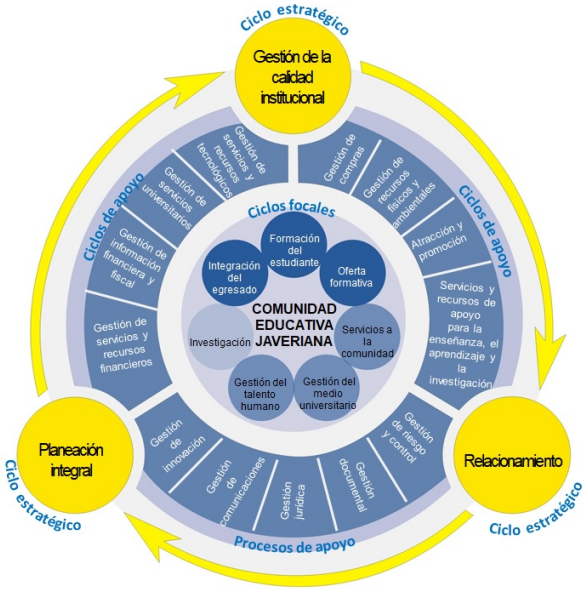

Figura 2.1: Modelo de procesos (Pontificia Universidad Javeriana Cali, 2021).

Dentro de los ciclos focales se encuentran los siguientes ciclos: formación del estudiante, oferta formativa, servicios a la comunidad, gestión del medio universitario, gestión del talento humano, investigación e integración del egresado.

El objetivo del ciclo de formación al estudiante es formar integralmente personas para que sobresalgan por su alta calidad humana, ética, académica, profesional y responsabilidad social

(Pontificia Universidad Javeriana Cali, 2021).

El ciclo de formación al estudiante se divide en los siguientes procesos: Admisión, Matrícula, Enseñanza-Aprendizaje, Gestión Académica-Administrativa y Grados como lo muestra la Figura 2.2.

Figura 2.2: Ciclo de formación al estudiante (Pontificia Universidad Javeriana Cali, 2021).

El objetivo de los procesos académico administrativos es proporcionar el soporte administrativo para facilitar el desarrollo de las actividades relacionadas con la formación del estudiante (Pontificia Universidad Javeriana Cali, 2021).

Modelos de lenguaje grandes (LLMs)

Los modelos de lenguaje grandes (Large Language Models, LLMs) son arquitecturas de aprendizaje profundo entrenadas con enormes volúmenes de datos textuales, lo que les permite comprender, generar y manipular lenguaje natural de manera contextual. Modelos como GPT (OpenAI), DeepSeek, LLaMA (Meta), Mistral y Falcon han demostrado capacidades sobresalientes para tareas como clasificación, resumen, traducción, generación de contenido y recuperación de información (Zhang et al., 2024). Para la ejecución local de estos modelos en entornos institucionales, existen herramientas de gestión y despliegue como Ollama, que actúan como *runtime* y permiten descargar, configurar y servir modelos de código abierto sin depender de servicios en la nube (Dorca Josa and Bleda-Bejar, 2024).

Su funcionamiento se apoya en la arquitectura *Transformer*, introducida por Vaswani et al. (2017), una red neuronal profunda diseñada para procesar secuencias de texto mediante mecanismos de *self-attention*, que permiten estimar la relación entre cada token y el resto del contexto. Esta

arquitectura continúa siendo la base de familias de modelos ampliamente utilizados, entre ellas GPT, LLaMA, Mistral, Qwen y DeepSeek.

La oferta de modelos abiertos ha evolucionado con rapidez en los últimos años. En julio de 2024, Meta publicó la familia LLaMA 3 en versiones de 8B, 70B y 405B parámetros (Grattafiori et al., 2024). Meses después, DeepSeek-AI presentó DeepSeek-V3, un modelo de 671B parámetros basado en una arquitectura *Mixture-of-Experts* (MoE), donde solo una parte de los parámetros se activa en cada inferencia (DeepSeek-AI, 2024). A comienzos de 2025, el mismo grupo introdujo DeepSeek-R1, orientado a tareas de razonamiento y entrenado mediante aprendizaje por refuerzo, mostrando que las mejoras de desempeño no dependen únicamente del aumento en el tamaño del modelo (DeepSeek-AI, 2025).

Estas variaciones en arquitectura y escala tienen consecuencias prácticas sobre los requerimientos de infraestructura. Las diferencias entre un modelo de 7B parámetros y otro de cientos de miles de millones pueden traducirse en demandas muy distintas de memoria, capacidad de cómputo y consumo energético (Samsi et al., 2023). Por esta razón, la selección de un modelo no puede analizarse de forma aislada del hardware disponible para ejecutarlo.

LLMs locales

En este trabajo se utiliza el término **LLMs locales** para referirse a modelos de lenguaje grandes ejecutados en infraestructura administrada por la propia institución (servidores *on-premise* o máquinas virtuales privadas), sin que las consultas, los documentos o las respuestas generadas viajen a servicios de terceros como OpenAI, Anthropic o Google. Esta categoría se diferencia tanto del consumo de LLMs vía API en la nube como del uso de modelos comerciales hospedados, y depende de tres elementos: (i) un modelo de pesos abiertos descargable publicado en repositorios como Hugging Face, (ii) un *runtime* de inferencia que cargue el modelo y lo exponga como servicio (por ejemplo Ollama, vLLM o llama.cpp), y (iii) hardware con suficiente memoria de GPU para mantener el modelo en VRAM durante la inferencia. La adopción de LLMs locales es relevante en escenarios con datos sensibles porque la información permanece dentro del perímetro institucional y queda sujeta a las políticas de seguridad y a los marcos normativos de la organización (Dorca Josa and Bleda-Bejar, 2024; Dennstädt et al., 2025).

Cuantización

Cuando un modelo se entrena, sus parámetros se almacenan con números en punto flotante de 16 o 32 bits (FP16 o FP32), un nivel de precisión que es útil durante el entrenamiento pero que resulta sobredimensionado para la inferencia. La cuantización aprovecha esa diferencia al representar cada parámetro con menos bits, generalmente enteros de 8 o 4 bits y en algunos casos hasta 2 bits, lo que reduce de manera proporcional la cantidad de memoria que el modelo ocupa cuando se carga en la GPU. En un modelo originalmente almacenado en FP16, la cuantización a 4 bits puede reducir el tamaño aproximado a una cuarta parte del original, con un efecto poco perceptible sobre la calidad de las respuestas en tareas de generación de proposito general (Dettmers et al., 2022; Frantar et al., 2023; Lin et al., 2024).

La diferencia se vuelve más clara al mirar un caso concreto. Un modelo de 7 mil millones de parámetros (7B) almacenado en FP16 ocupa cerca de 14 GB, una cantidad que supera la memoria disponible en GPU de consumo con 12 GB de VRAM. Si ese mismo modelo se cuantiza a 4 bits, su tamaño suele reducirse a un rango de entre 4 y 5 GB, lo que deja memoria disponible para el contexto de inferencia y evita depender de la RAM del sistema, una situación que normalmente degrada el rendimiento. Esta reducción ha facilitado que modelos de lenguaje relativamente grandes puedan ejecutarse en infraestructura local de menor capacidad computacional, sin depender de aceleradores especializados de centro de datos. Entre los formatos más utilizados para distribuir modelos cuantizados se encuentran GGUF, usado en el ecosistema de llama.cpp, y GPTQ (Frantar et al., 2023). En esquemas como AWQ, el proceso de cuantización busca conservar mejor el desempeño del modelo al priorizar los pesos más sensibles (Lin et al., 2024).

Agentes de Inteligencia Artificial

Los agentes de inteligencia artificial (IA) son sistemas de software capaces de percibir su entorno, razonar y ejecutar acciones de forma autónoma o asistida para alcanzar objetivos específicos (Russell and Norvig, 2021). Estos agentes pueden actuar de manera reactiva o proactiva, y utilizan diversas técnicas de IA, como machine learning, procesamiento de lenguaje natural o planificación automática, para tomar decisiones adaptativas en entornos dinámicos.

Según Lanham (2025), existen cuatro formas principales en las que un usuario puede interactuar con un modelo de lenguaje grande (LLM) como se puede observar en la Figura 2.3: mediante interacción directa, a través de un agente proxy (intermediario), un agente funcional o un agente autónomo. Estas modalidades representan distintos niveles de complejidad y autonomía en la ejecución de tareas, y se describen a continuación:

1. **Interacción directa**: En este tipo de interacción, el usuario formula una consulta y el modelo genera una respuesta únicamente a partir de la información aprendida durante su entrenamiento, sin acceder a fuentes externas ni utilizar herramientas adicionales. Un caso común ocurre al pedir a asistentes como ChatGPT o Claude la redacción de un correo o el resumen de un texto proporcionado directamente en la conversación. En estos escenarios, la respuesta se construye con base en el conocimiento interno del modelo y el contenido disponible en el intercambio con el usuario.

2. **Agente proxy (intermediario)**: En este tipo de interacción, el modelo actúa como intermediario entre el usuario y otro sistema, reformulando o ampliando la instrucción original para facilitar su procesamiento. Esto puede observarse en algunos generadores de imágenes integrados en plataformas como ChatGPT, Gemini o Copilot, donde una solicitud breve del usuario se transforma en un prompt más detallado antes de enviarse al modelo encargado de producir la imagen. Aunque esa reformulación suele ocurrir de manera transparente para el usuario, el nivel de detalle incorporado durante este paso puede influir en el resultado final.

3. **Agente funcional**: El modelo dispone de un conjunto definido de herramientas externas que puede utilizar para ejecutar determinadas acciones, aunque su uso permanece sujeto a la

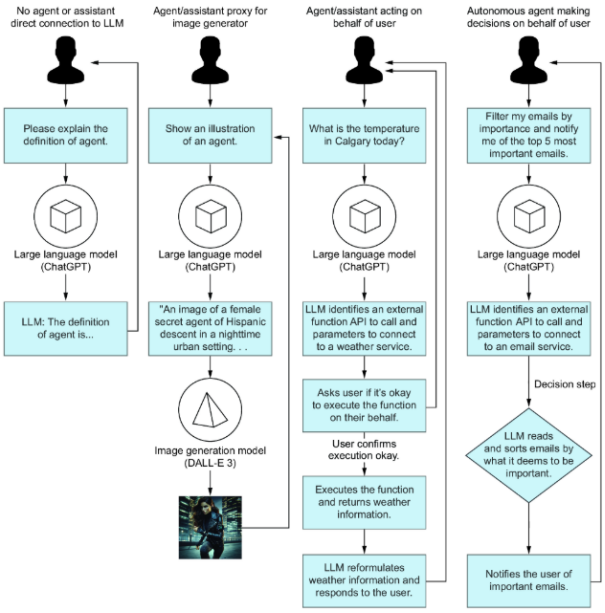

Figura 2.3: Diferencias entre las interacciones con modelos de lenguaje grandes (LLMs) mediante acción directa en comparación con el uso de agentes proxy, agentes funcionales y agentes autónomos (Lanham, 2025).

autorización del usuario. Este esquema es frecuente en asistentes de desarrollo como GitHub Copilot, Cursor o Claude Code. Por ejemplo, ante una instrucción como "ejecuta las pruebas y corrige los errores", el sistema puede sugerir acciones como ejecutar comandos en la terminal, revisar archivos del proyecto o proponer modificaciones en el código. Antes de aplicar estos cambios, el usuario decide qué acciones autoriza y cuáles prefiere descartar.

4. **Agente autónomo**: En este nivel de autonomía, el sistema recibe un objetivo general y ejecuta distintas acciones para alcanzarlo, mientras la intervención del usuario ocurre solo en momentos específicos o una vez finalizada la tarea. Un ejemplo puede observarse en agentes de operación continua orientados a la gestión de buzones de correo electrónico. Estos sistemas pueden clasificar correos según prioridad, sugerir respuestas para solicitudes frecuentes, derivar casos que requieren intervención humana y mantener un registro de las acciones realizadas. A medida que aumenta el nivel de autonomía, también aparecen retos relacionados con seguridad, trazabilidad y toma de decisiones. Al no existir una validación paso a paso, los mecanismos de control y las restricciones deben incorporarse desde el diseño del agente.

En el marco de este trabajo de grado, se desarrollaron prototipos correspondientes a dos de las categorías descritas: agentes funcionales y agentes autónomos. Los primeros operan como asistentes capaces de interactuar con funciones externas bajo aprobación del usuario, mientras que los segundos buscan ejecutar tareas de forma más independiente, tomando decisiones sobre el flujo

de trabajo según las condiciones del entorno. Esta distinción permitió evaluar las implicaciones técnicas, de seguridad y usabilidad entre ambos enfoques en contextos académico-administrativos.

Recuperación aumentada por generación (RAG)

La Recuperación Aumentada por Generación (*Retrieval-Augmented Generation*, RAG) fue propuesta por Lewis et al. (2020) como una forma de complementar el conocimiento del modelo con información proveniente de fuentes externas. En lugar de depender únicamente de lo aprendido durante el entrenamiento, el sistema incorpora documentos o fragmentos relevantes al momento de responder una consulta. Para ello, primero recupera información desde una base de conocimiento. Por ejemplo, archivos institucionales, bases de datos o repositorios documentales y luego envía ese material junto con la consulta al LLM, que genera la respuesta considerando ambos elementos. La revisión de Gao et al. (2023) describe distintas variantes de esta arquitectura, desde esquemas más simples, donde la recuperación ocurre antes de la generación, hasta enfoques que incorporan etapas adicionales como reordenamiento de resultados (*re-ranking*), reformulación de consultas o flujos modulares de varias fases.

En contextos institucionales, RAG resulta especialmente útil porque permite separar el conocimiento general del modelo de la información propia de la organización. Esto significa que las respuestas pueden actualizarse al modificar los documentos indexados, sin necesidad de reentrenar el modelo ni transferir información institucional a servicios externos. En una universidad, esta característica puede facilitar escenarios como consultas sobre normativas, procesos académicos o información administrativa, donde los contenidos cambian con cierta frecuencia. Por esta razón, RAG se considera uno de los puntos de partida más viables para explorar el uso de LLMs locales en entornos con restricciones de privacidad y constituye la base del primer prototipo desarrollado en este trabajo (Capítulo 4).

Transformación Digital en Educación Superior

La transformación digital en educación superior se entiende hoy como un fenómeno estructural que reconfigura los procesos institucionales más allá de la simple adopción de herramientas, e impacta a estudiantes, docentes y personal administrativo (Cuzcano et al., 2026). Un aspecto clave de esta transformación es la automatización inteligente, que combina IA y automatización de procesos para evolucionar hacia organizaciones más ágiles centradas en el usuario.

La incorporación de agentes de IA en procesos académico-administrativos se alinea con las tendencias globales de digitalización, respondiendo a la necesidad de instituciones más competitivas, innovadoras y resilientes frente a los desafíos del entorno actual.

2.2. Estado del arte

En los últimos años, múltiples instituciones educativas y gubernamentales han comenzado a implementar soluciones de automatización y agentes de inteligencia artificial (IA) en sus procesos

administrativos y académicos. Estos esfuerzos reflejan una tendencia creciente hacia la transformación digital, en respuesta a la necesidad de mayor eficiencia, precisión y escalabilidad en los servicios educativos. A continuación se presentan diferentes casos de uso de agentes de inteligencia artificial en el sector educativo, en cada uno de ellos se expone el problema que presentaban las instituciones y los beneficios que les trajo implementar la solución.

Universidad de Münster – UniGPT (Alemania)

Radas et al. (2024) documentan la creación de UniGPT, un agente de IA tipo chatbot implementado por la Universidad de Münster para responder consultas académicas internas. El sistema fue desarrollado utilizando modelos open-source como Mistral y LLaMA 2, y se desplegó en servidores locales de la universidad con una arquitectura multi-GPU. El objetivo principal del proyecto fue lograr soberanía digital, es decir, reducir la dependencia de plataformas como OpenAI y garantizar el control sobre los datos institucionales. Los autores destacan que, si bien la solución permitió preservar la privacidad y personalizar el comportamiento del agente, el desarrollo y mantenimiento del sistema requirió personal técnico altamente calificado y recursos de cómputo considerables (Radas et al., 2024).

Universidad de Andorra – LLMs locales para la protección de datos

Dorca Josa and Bleda-Bejar (2024) presentan un estudio de caso en la Universidad de Andorra, donde se implementó una infraestructura local de LLMs utilizando herramientas como Ollama y HuggingFace para permitir el uso de modelos como GPT4All y LLaMA por parte del personal docente y administrativo. El proyecto nació como respuesta a preocupaciones sobre privacidad institucional y protección de datos personales, y tuvo como principio rector que los datos no debían salir del entorno de la universidad. El estudio evidencia que la solución fue exitosa en términos de seguridad y control, pero que las limitaciones de hardware y la necesidad de capacitación técnica representaron retos importantes para la escalabilidad.

Universidad Autónoma de Barcelona – Asistente inteligente para gestión académica

La Universidad Autónoma de Barcelona (UAB) implementó un asistente conversacional basado en IA para responder consultas frecuentes de estudiantes sobre matrículas, horarios, requisitos y trámites académicos. El agente, integrado a la página web institucional y canales de mensajería, logró atender más del 60 % de las consultas sin intervención humana durante el primer semestre de uso, reduciendo la carga del personal administrativo y mejorando la satisfacción estudiantil. Este resultado fue documentado por González and Martínez (2023) en un estudio de caso publicado en la *Revista Iberoamericana de Educación Superior*, basado en datos operativos reportados por la propia institución.

Universidad de Helsinki – Automatización del proceso de gestión de prácticas estudiantiles

La Universidad de Helsinki implementó un sistema basado en RPA y agentes conversacionales para automatizar el proceso de gestión de prácticas profesionales. El sistema automatiza tareas como la validación de convenios, el registro de informes de avance y la notificación de fechas clave tanto a estudiantes como a tutores académicos. Lindholm and Kallio (2023), en un artículo publicado en el *Journal of Higher Education Innovation and Technology*, reportan una reducción del 45 % en el tiempo promedio de tramitación de cada práctica y un incremento significativo en la satisfacción estudiantil, con base en métricas operativas registradas durante el primer año de implementación.

Universidad Nacional de Singapur – Agente de IA para gestión de solicitudes estudiantiles

La Universidad Nacional de Singapur (NUS) integró un agente de IA desarrollado sobre lenguaje natural para responder, clasificar y canalizar solicitudes estudiantiles recibidas vía correo electrónico y formularios web. Según Tan et al. (2023), documentado en el *International Journal of Educational Technology Integration*, el agente fue entrenado con más de 50.000 interacciones previas y logró redirigir correctamente más del 80 % de las solicitudes sin intervención humana, reduciendo los tiempos de respuesta de días a horas. Los autores señalan que esta tasa fue medida sobre un periodo de seis meses de operación en producción.

Despliegues clínicos de LLMs locales (2025)

Más allá del sector educativo, en 2025 se consolidó la adopción de LLMs locales en instituciones que manejan información altamente sensible, particularmente en el sector salud. Dennstädt et al. (2025) documentan, en *npj Digital Medicine*, un marco de implementación de LLMs en hospitales que balancea cuatro dimensiones simultáneamente: control sobre los datos clínicos, colaboración entre equipos médicos y de TI, costos de operación y seguridad. El trabajo concluye que la opción *on-premise* es preferible cuando los datos no pueden salir del perímetro institucional –como ocurre con expedientes clínicos protegidos por HIPAA y normativas equivalentes– y propone arquitecturas modulares que combinan modelos abiertos cuantizados con interfaces seguras de consumo. La pertinencia de este caso para el presente trabajo radica en que las tensiones que identifica el estudio (control vs. costo, autonomía vs. esfuerzo de mantenimiento) son las mismas que aparecen al evaluar el despliegue de LLMs locales en el contexto académico-administrativo.

Hospital Universitario de Magdeburgo (Alemania) – LLM on-premise con evaluación de usuarios (2026)

Grünig et al. (2026) reportan, en *JMIR AI*, la implementación y evaluación de un LLM ejecutado completamente *on-premise* en el Hospital Universitario de Magdeburgo. El sistema utiliza el modelo abierto **LLaMA 3.2-Vision (90B)** desplegado sobre un clúster local de tres GPUs y está

expuesto al personal mediante una interfaz interna, sin enviar datos a servicios externos. La evaluación es lo que distingue a este trabajo de los antecedentes previos: en lugar de quedarse en una descripción técnica, los autores aplicaron una encuesta transversal a **91 usuarios** (personal clínico y administrativo) y reportaron resultados cuantitativos sobre la experiencia de uso. Aproximadamente el 70 % de los encuestados calificó la velocidad y confiabilidad del sistema como "muy alta" y cerca del 60 % calificó la calidad de las respuestas como "alta"; muchos usuarios reportaron además un ahorro mediano de tiempo administrativo del orden de 30 minutos por día gracias al asistente. Los autores subrayan que la decisión de autoalojar el modelo fue determinante para cumplir con el GDPR sin necesidad de un acuerdo de procesamiento de datos con un proveedor externo. Este caso es particularmente relevante para el presente trabajo porque combina los tres elementos que aquí se evalúan: ejecución local, evaluación de usuarios y soberanía de datos y aporta evidencia empírica reciente (2026) de que la modalidad on-premise es viable en un contexto universitario real.

Análisis del estado del arte

La revisión de antecedentes muestra que el interés por utilizar agentes de IA y modelos de lenguaje en procesos académico-administrativos ya tienen aplicaciones concretas en distintos contextos institucionales. También deja ver diferencias importantes en la forma en que estas iniciativas abordan aspectos como privacidad, infraestructura, alcance funcional y evaluación del desempeño.

Los casos de UniGPT (Radas et al., 2024) y de la Universidad de Andorra (Dorca Josa and Bleda-Bejar, 2024) son los antecedentes más cercanos al enfoque de este trabajo, particularmente por el interés en ejecutar modelos dentro de infraestructura institucional y mantener control sobre la información procesada. Una preocupación similar aparece en el sector salud: el caso clínico documentado por Dennstädt et al. (2025) y la experiencia del Hospital Universitario de Magdeburgo reportada por Grünig et al. (2026) muestran escenarios donde la sensibilidad de los datos limita el uso de servicios externos y favorece despliegues *on-premise*. En conjunto, estos antecedentes sugieren que la necesidad de mantener información sensible dentro del perímetro institucional no es exclusiva del ámbito universitario.

Aun así, los trabajos revisados tienden a concentrarse en problemas específicos. Münster y Andorra se enfocan principalmente en asistentes conversacionales orientados a consultas documentales; Dennstädt et al. (2025) profundiza en aspectos de implementación y gobernanza; y Grünig et al. (2026) prioriza la experiencia de usuario en un entorno hospitalario. Ninguno de estos casos evalúa varios escenarios de uso sobre una misma infraestructura ni incorpora simultáneamente pruebas de carga y estimaciones económicas asociadas al despliegue local.

Una situación diferente aparece en experiencias como las de la UAB (González and Martínez, 2023), la Universidad de Helsinki (Lindholm and Kallio, 2023) y la NUS (Tan et al., 2023). Aunque reportan mejoras en distintos procesos universitarios, sus implementaciones dependen de servicios comerciales o modelos ejecutados en la nube. Esto puede representar una dificultad en escenarios donde se trabaja con documentos institucionales o información sensible, ya que parte del procesamiento ocurre fuera de la infraestructura de la organización. En el contexto de este trabajo, esta diferencia hizo relevante examinar hasta qué punto un despliegue local podía ofrecer resultados

útiles sin necesidad de transferir la información a servicios externos.

La revisión también muestra un vacío en torno a evaluaciones que integren simultáneamente dimensiones técnicas, económicas y de privacidad sobre un mismo conjunto de prototipos. Dentro de los antecedentes revisados, esta combinación aparece de manera fragmentada: algunos estudios se concentran en el desempeño técnico, otros en adopción institucional y otros en gobernanza de datos. El presente trabajo buscó articular estas dimensiones mediante la implementación y evaluación de tres casos de uso sobre infraestructura local, tomando como referencia el contexto específico de la Pontificia Universidad Javeriana Cali.

Más que mostrar una única dirección de adopción, los antecedentes revisados indican que el uso institucional de agentes basados en LLMs depende del equilibrio entre restricciones de privacidad, capacidades de infraestructura y necesidades concretas de cada organización. Ese panorama ayuda a ubicar el alcance de este proyecto y el tipo de preguntas que buscó responder frente a la literatura existente.

Capítulo 3

Requerimientos para la adopción de LLMs locales en entornos institucionales

3.1. Identificación de requerimientos técnicos para la ejecución local de LLMs

El despliegue de modelos de lenguaje de gran escala (LLMs) en entornos institucionales requiere una planificación detallada en términos de hardware, software y costos. Este capítulo presenta un análisis comparativo de diversos modelos de lenguaje *open source* y ejecutables localmente, con el fin de establecer los requerimientos técnicos mínimos para su funcionamiento, y así facilitar su adopción en contextos académicos con restricciones de privacidad de datos. Antes de comparar modelos conviene precisar los tres componentes de hardware que aparecen como variables en el resto del capítulo. La **GPU** (*Graphics Processing Unit*) es el componente que ejecuta la inferencia del modelo: su memoria dedicada (**VRAM**, *Video RAM*) debe ser suficiente para albergar los pesos del modelo y el contexto activo. La **CPU** (*Central Processing Unit*) coordina el flujo del programa, atiende solicitudes y maneja la entrada/salida; su capacidad importa, pero no es el cuello de botella en inferencia con LLMs. La **RAM** del sistema se utiliza para cargar el modelo desde disco antes de moverlo a la GPU y para sostener componentes auxiliares (servidor web, base vectorial, orquestador). No obstante cuando un modelo no cabe completo en VRAM, parte de los pesos se mantiene en RAM y se transfiere por demanda, lo que reduce el rendimiento. Por esta razón la VRAM es la variable más restrictiva en la selección de hardware para LLMs locales. Se seleccionaron seis familias de modelos representativas del ecosistema *open source* disponible en 2025–2026, considerando que cubren distintos rangos de tamaño y casos de uso. La Tabla 3.1 resume cada familia y la organización que la desarrolla.

Tabla 3.1: Familias de LLMs evaluadas y sus organizaciones desarrolladoras

Familia	**Organización**	**Características**
LLaMA 3.x	Meta AI	Familia base de pesos abiertos más adoptada en el ecosistema *open source*; cubre desde 8B hasta 405B parámetros.
Mistral	Mistral AI (Francia)	Modelos densos y *Mixture-of-Experts* con buen rendimiento por parámetro; popular en despliegues europeos.
Qwen 2.5/3.x	Alibaba Cloud	Multilingüe (incluye español) y con variantes especializadas en código y razonamiento.
Phi-4	Microsoft Research	Modelos compactos (14B) optimizados para tareas de razonamiento, pensados para despliegue en hardware modesto.
Gemma	Google DeepMind	Versión abierta derivada de la familia Gemini; tamaños 2B y 7B.
DeepSeek	DeepSeek AI (China)	Modelos de gran escala (67B–671B) con énfasis en eficiencia de inferencia.

Para cada familia se evaluaron los requerimientos de infraestructura, el desempeño en inferencia y las capacidades de optimización mediante cuantización descritas en el Capítulo 2.

3.1.1. Recolección de información

La información se obtuvo a partir de tres fuentes principales:

1. **Documentación oficial**: Meta (LLaMA 3), Mistral AI, Alibaba (Qwen), Microsoft (Phi-4), Google (Gemma) y DeepSeek. Estas fuentes proporcionan especificaciones técnicas, tamaños de modelos y capacidades de contexto.

2. **Benchmarks y reportes técnicos**: Hugging Face, estudios sobre inferencia y documentación de fabricantes como NVIDIA (RTX 4090, A100, H100, L40S), incluyendo métricas de rendimiento en inferencia local.

3. **Literatura científica al momento de desarrollar el trabajo de grado**: Estudios sobre despliegue de LLMs en entornos académicos, como UniGPT (Radas et al., 2024) y el caso de la Universidad de Andorra (Dorca Josa and Bleda-Bejar, 2024), así como investigaciones sobre requerimientos de infraestructura y riesgos asociados.

Con base en estas fuentes, se construyó una matriz comparativa que sintetiza los requerimientos técnicos mínimos y recomendados para modelos entre 7B y más de 100B parámetros.

3.1.2. Matriz comparativa de requerimientos técnicos (2025–2026)

La Tabla 3.2 reúne configuraciones representativas de modelos abiertos disponibles entre 2025 y 2026, organizadas según variables relevantes para planear una implementación local. Se incluyen aspectos como el número de parámetros, el tamaño aproximado en disco, la memoria de video (VRAM) mínima y recomendada, la memoria RAM del sistema, el tipo de hardware requerido, almacenamiento, rendimiento estimado en tokens por segundo, soporte de cuantización, ventana de contexto, consumo aproximado de GPU durante inferencia y complejidad de despliegue.

El consumo energético se incorpora como criterio de comparación porque influye de manera directa en los costos de operación de una infraestructura local. Por ejemplo, una GPU con un consumo cercano a 300 W operando de forma continua puede superar los 2.600 kWh anuales únicamente en tareas de inferencia, sin considerar el resto de componentes del servidor. En entornos institucionales, esta diferencia adquiere relevancia cuando se evalúa la viabilidad de mantener modelos ejecutándose de forma permanente.

La comparación permite observar algunas diferencias importantes entre escalas de modelos. Configuraciones entre 7B y 14B parámetros suelen ejecutarse en GPUs de consumo con al menos 16 GB de VRAM y presentan consumos cercanos al rango de 150 a 300 W. A medida que aumenta el número de parámetros, también crecen los requerimientos de memoria y energía. En modelos superiores a 30B parámetros, se vuelve más frecuente la necesidad de GPUs orientadas a centros de datos o aceleradores de alto desempeño, con demandas de VRAM entre 48 y 80 GB y consumos que pueden superar los 700 W por GPU en escenarios de inferencia continua.

Tabla 3.2: Matriz comparativa de requerimientos técnicos para LLMs locales (2025–2026). El consumo de GPU corresponde al rango típico durante inferencia continua según las TDP publicadas por el fabricante de la GPU asociada a cada categoría de hardware.

Modelo	Parám.	Tamaño	VRAM (min/rec)	RAM	Hardware	Storage	Tokens/s	Cuant.	Contexto	Consumo (W)	Complejidad
LLaMA 3 8B	8B	16 GB	8 / 16 GB	32 GB	GPU consumer	50 GB	40–120	4/8-bit	8K–128K	150–300	Baja
LLaMA 3 70B	70B	140 GB	48 / 80 GB	128 GB	Enterprise	200 GB	10–40	4/8-bit	8K–128K	400–700	Alta
Mistral Small (7B)	7B	14 GB	8 / 16 GB	32 GB	Consumer	40 GB	50–150	4/8-bit	32K	150–300	Baja
Mistral Large	¿100B	¿200 GB	80+ GB	256 GB	Enterprise	500 GB	5–20	4-bit	32K–128K	1.500+	Muy alta
Qwen 2.5 7B	7B	14 GB	8 / 16 GB	32 GB	Consumer	40 GB	40–130	4/8-bit	32K–128K	150–300	Baja
Qwen 2.5 72B	72B	150 GB	48 / 80 GB	128 GB	Enterprise	250 GB	10–35	4-bit	32K–128K	400–700	Alta
Phi-4	14B aprox.	28 GB	16 / 24 GB	64 GB	Consumer+	80 GB	30–100	4/8-bit	16K–128K	200–350	Media
Gemma 7B	7B	13 GB	8 / 16 GB	32 GB	Consumer	40 GB	40–120	4/8-bit	8K–32K	150–300	Baja
DeepSeek 67B	67B	130 GB	48 / 80 GB	128 GB	Enterprise	200 GB	10–30	4-bit	32K–128K	400–700	Alta

3.1.3. Análisis de los requerimientos identificados

Antes de revisar los resultados de la matriz comparativa, conviene aclarar un concepto que influye directamente en los requerimientos de hardware. La mayoría de los modelos incluidos corresponden a *modelos densos*, es decir, arquitecturas en las que todos los parámetros participan durante cada inferencia. Esto contrasta con modelos basados en arquitecturas *Mixture-of-Experts* (MoE), como DeepSeek-V3, donde solo una parte de los parámetros se activa en cada ejecución mientras el resto permanece inactivo. La diferencia tiene implicaciones prácticas: en un modelo denso, la memoria necesaria depende del tamaño completo del modelo, mientras que en un MoE la carga efectiva puede ser menor.

Al revisar la Tabla 3.2, la memoria de video (VRAM) aparece como una de las restricciones más determinantes para la ejecución local. Un modelo de 7B parámetros en precisión FP16 ocupa cerca de 14 GB; uno de 14B se aproxima a 28 GB y uno de 70B supera los 140 GB. Mientras el modelo puede mantenerse completamente en la memoria de la GPU, la generación de respuestas ocurre de forma fluida. El comportamiento cambia cuando la VRAM resulta insuficiente: parte de los pesos debe transferirse desde la memoria RAM del sistema, un proceso que incrementa la latencia y afecta el rendimiento en escenarios interactivos. Esto explica por qué modelos superiores a 30B parámetros suelen quedar fuera del rango de equipos con GPU de consumo entre 16 y 24 GB de VRAM.

En este contexto, la cuantización reduce de los requerimientos de memoria. Al representar los pesos con menor precisión. Por ejemplo, 4 bits en lugar de FP16, el tamaño del modelo puede disminuir entre tres y cuatro veces. Un modelo de 7B pasa de ocupar aproximadamente 14 GB a un rango cercano entre 4 y 5 GB, mientras que uno de 14B puede reducirse a alrededor de 8 o 9 GB. Esta diferencia deja espacio para el contexto activo de la conversación dentro de GPUs de 16 GB y amplía el conjunto de modelos que pueden ejecutarse localmente. Aunque la pérdida de calidad puede observarse en algunos *benchmarks*, suele mantenerse dentro de un margen aceptable para tareas como consulta de documentos, transcripción asistida o renombramiento de información. Por esta razón, muchos de los modelos distribuidos en formatos como GGUF se publican ya cuantizados.

El rendimiento percibido también cambia según el tamaño del modelo y el hardware disponible. La columna de *tokens/s* de la Tabla 3.2 muestra que modelos de 7B ejecutados sobre GPU de consumo pueden generar entre 40 y 150 tokens por segundo, mientras que configuraciones cercanas a 70B suelen ubicarse entre 10 y 40 tokens por segundo incluso sobre GPUs orientadas a centros de datos o aceleradores de alto desempeño. Para respuestas de alrededor de 300 tokens. Esta diferencia puede traducirse en tiempos cercanos a 2–8 segundos en modelos de menor tamaño frente a rangos de 8–30 segundos en configuraciones más grandes, por lo cual, en aplicaciones conversacionales, ese cambio suele percibirse con facilidad. En cambio, en procesos asincrónicos como la transcripción de audios o el renombramiento masivo de documentos la latencia individual pierde relevancia, ya que el resultado no se espera de forma inmediata.

El consumo energético introduce otra variable que suele pasar desapercibida cuando la discusión se centra únicamente en capacidad de cómputo. Las configuraciones entre 7B y 14B operan normalmente entre 150 y 300 W, un rango comparable al de un equipo de escritorio. En modelos cercanos a 70B, el consumo puede duplicarse o incluso triplicarse dependiendo de la GPU utilizada. Cuando la operación es continua, esta diferencia se acumula en el tiempo y termina reflejándose en costos eléctricos relevantes para una institución.

3.1.4. Rangos típicos de infraestructura por tamaño de modelo

La matriz comparativa permite agrupar los modelos en cuatro rangos de infraestructura según sus requerimientos de memoria, capacidad de procesamiento y complejidad operativa. Esta clasificación se construyó tomando como referencia escenarios institucionales donde existe infraestructura de TI propia y se espera atender volúmenes moderados de uso, como consultas sobre documentos,

transcripciones o tareas de clasificación de información similares a las evaluadas en este trabajo.

- **7B–8B** — Requiere normalmente una GPU con entre 8 y 16 GB de VRAM y alrededor de 32 GB de RAM del sistema. Este rango suele ser suficiente para tareas como recuperación aumentada por generación (RAG), transcripción o clasificación documental, manteniendo tiempos de respuesta interactivos para grupos pequeños de usuarios concurrentes. Los prototipos desarrollados en este trabajo se implementaron sobre esta escala de modelos.
- **13B–14B** — Generalmente demanda una GPU con 16–24 GB de VRAM y cerca de 64 GB de RAM. Frente a configuraciones más pequeñas, ofrece mejoras en tareas que requieren respuestas extensas, manejo multilingüe o mayor consistencia contextual, sin depender todavía de GPUs orientadas a centros de datos o aceleradores de alto desempeño.
- **70B** — En este rango suelen requerirse múltiples GPUs trabajando en paralelo o aceleradores de centro de datos con capacidades cercanas a 80 GB de VRAM. La complejidad de despliegue y operación aumenta de forma considerable, por lo que este tipo de configuraciones tiende a reservarse para escenarios donde se busca un mayor nivel de razonamiento o asistentes de propósito más general.
- $>$ 100**B** — Normalmente implica entornos distribuidos con interconexiones de alto ancho de banda y una capacidad de cómputo difícil de sostener para una sola institución. En estos casos, resulta más frecuente el uso de infraestructura compartida o servicios especializados de nube.

3.1.5. Frameworks y ecosistema de despliegue

La ejecución local de un LLM no depende únicamente del modelo seleccionado, sino también del entorno utilizado para cargarlo, gestionar memoria y exponer servicios de inferencia. Durante 2025 y 2026, distintos frameworks han sido utilizados para facilitar estas tareas, con diferencias en facilidad de uso, capacidad de personalización y rendimiento.

- **Ollama** (Ollama, 2025) — Permite ejecutar modelos cuantizados en formato GGUF mediante un servidor local que expone una API compatible con OpenAI. Su adopción se ha extendido en escenarios de prototipado y despliegues pequeños debido a que simplifica tareas como la descarga del modelo, su carga en GPU y la configuración inicial.
- **llama.cpp** (Gerganov and llama.cpp contributors, 2025) — Implementación en C++ orientada a la inferencia eficiente en CPU y GPU mediante modelos cuantizados. Es utilizada en contextos donde se requiere un mayor control sobre formatos de cuantización o cuando el hardware disponible no incluye GPU dedicada. Herramientas como Ollama se apoyan parcialmente en este ecosistema.
- **vLLM** (vLLM Project, 2025) — Framework orientado a escenarios de producción con múltiples solicitudes concurrentes. Emplea técnicas como *PagedAttention* para optimizar el uso de memoria durante la inferencia y mantener tiempos de respuesta más estables bajo carga.

- **Transformers de Hugging Face** (Hugging Face, 2025) — Biblioteca de Python utilizada como base para implementar y experimentar con una variedad de modelos abiertos. Aunque es flexible, también requiere una configuración más detallada del entorno de ejecución, incluyendo manejo de memoria, tokenización y servicios de inferencia.

Para los prototipos desarrollados en este trabajo se utilizó Ollama, principalmente porque permitió implementar los casos de uso sin requerir configuraciones complejas ni optimizaciones adicionales. La decisión también respondió a la necesidad de priorizar un entorno reproducible con menores requerimientos de configuración técnica dentro del contexto institucional evaluado. Los criterios específicos de implementación se describen con mayor detalle en el Capítulo 4.

3.1.6. Conclusiones técnicas

Los resultados del análisis sugieren que la ejecución local de LLMs puede ser factible en instituciones con características similares a la Pontificia Universidad Javeriana Cali, siempre que los casos de uso puedan resolverse con modelos en el rango de 7B a 14B parámetros y exista disposición para trabajar con versiones cuantizadas. Bajo estas condiciones, los requerimientos de hardware permanecen dentro de capacidades cercanas a una estación de trabajo equipada con GPU entre 16 y 24 GB de VRAM, sin depender necesariamente de infraestructura de centro de datos.

Esta posibilidad se relaciona con el tipo de tareas evaluadas en el trabajo. Los tres casos de uso considerados, consultas sobre documentos, transcripción asistida y clasificación o renombramiento de archivos, corresponden a procesos acotados desde el punto de vista contextual. En escenarios de este tipo, las diferencias observadas entre modelos pequeños y configuraciones mucho más grandes no siempre justifican el incremento en requerimientos de memoria, consumo energético y costo de infraestructura. La cuantización también amplía el margen de despliegue local al reducir el uso de memoria, permitiendo ejecutar modelos de 7B y 14B dentro de GPUs de consumo sin afectar el comportamiento esperado para estas tareas.

Por un lado, en configuraciones de este tamaño, el principal límite operativo suele aparecer cuando aumenta la cantidad de solicitudes simultáneas. En muchos casos, el reto deja de estar en ejecutar el modelo y pasa a sostener tiempos de respuesta aceptables para varios usuarios al mismo tiempo. Esto sugiere que, ante un aumento de demanda, puede resultar más práctico replicar instancias del mismo modelo que migrar inmediatamente hacia arquitecturas de mayor tamaño.

Por otro lado, las necesidades cambian cuando se requieren tareas de propósito general, respuestas extensas o procesos que demandan un mayor nivel de razonamiento. En esos casos, modelos de 70B parámetros o superiores pueden ofrecer ventajas, aunque acompañadas de incrementos en memoria, consumo energético y complejidad operativa. Para instituciones con restricciones de infraestructura, una combinación entre modelos locales para información sensible y servicios externos para necesidades puntuales puede representar una alternativa más manejable que reproducir localmente capacidades similares a las ofrecidas por grandes proveedores comerciales.

3.1.7. Recomendaciones

A partir del análisis anterior, se proponen las siguientes recomendaciones operativas para una institución que vaya a adoptar LLMs locales:

- Partir del caso de uso, no del modelo: identificar primero qué tarea se quiere automatizar y qué calidad de respuesta es aceptable, y a partir de eso seleccionar el tamaño de modelo más pequeño que cumpla. En la mayoría de tareas académico-administrativas analizadas, un modelo 7B–14B cuantizado a 4 bits es suficiente.
- Dimensionar el hardware sobre la VRAM, no sobre la cantidad de núcleos de CPU. Una GPU de 16–24 GB de VRAM cubre los modelos del rango recomendado; CPU y RAM se ajustan después en función de los servicios auxiliares (servidor web, base vectorial, orquestador).
- Estandarizar sobre un único *runtime* (en este trabajo, Ollama) para que la operación, el monitoreo y la actualización de modelos sigan un procedimiento uniforme entre prototipos.
- Considerar una arquitectura híbrida cuando aparezcan casos de uso que excedan el rango 7B–14B: lo que permitirá mantener los datos sensibles en infraestructura local y delegar las tareas puntuales que requieran modelos mayores a un proveedor en la nube con cláusula contractual de no entrenamiento.

3.2. Análisis de costos de implementación y mantenimiento de agentes de IA con LLMs locales

En esta sección se comparan los costos estimados de despliegue, operación y mantenimiento entre dos enfoques: la ejecución local (*on-premise*) y el uso de soluciones basadas en la nube (*cloud-based*).

Los valores se estructuran siguiendo la distinción contable entre **CAPEX** (*Capital Expenditure*, inversión inicial en activos como hardware) y **OPEX** (*Operating Expenditure*, gastos recurrentes de operación) (Investopedia, 2024a,b). Esta separación es la que aplican habitualmente los análisis de costo total de propiedad (TCO) en infraestructura de TI y permite comparar la modalidad local, donde el grueso del costo es CAPEX, con la modalidad en la nube,donde casi todo el costo es OPEX. Las cifras de hardware se basaron en precios de catálogo de fabricantes y de distribuidores en Colombia consultados durante el primer trimestre de 2026 (NVIDIA, 2023; Dennstädt et al., 2025; Samsi et al., 2023); las tarifas de la nube se tomaron de las páginas oficiales de precios de OpenAI, Anthropic y Google en el mismo periodo. Conviene anotar que tanto los precios de hardware como las tarifas de los proveedores de IA en la nube cambian con frecuencia (las tarifas de inferencia de modelos comerciales han bajado de forma sostenida desde 2023), por lo que las cifras de esta sección deben leerse como una fotografía del momento del análisis y no como valores estables a mediano plazo.

3.2.1. Estructura de costos evaluados

Las categorías consideradas para el análisis de costos se agruparon en tres componentes principales, los cuales permiten evaluar de manera estructurada los diferentes escenarios de implementación:

- **Inversión inicial (CAPEX):** Corresponde a los costos asociados a la adquisición de la infraestructura necesaria para la ejecución local de LLMs. Esta categoría incluye la compra de hardware como unidades de procesamiento gráfico (GPU), procesadores (CPU), memoria RAM, sistemas de almacenamiento de alto rendimiento y mecanismos de respaldo eléctrico requeridos para garantizar la operación continua del sistema.
- **Costos operativos (OPEX):** Incluye los gastos recurrentes derivados del funcionamiento y mantenimiento de la infraestructura local, tales como el consumo de energía eléctrica, las actividades de mantenimiento preventivo y correctivo, el soporte técnico, el reemplazo de componentes por desgaste y la actualización periódica de los modelos y herramientas utilizadas.
- **Costos basados en la nube:** Comprende los gastos asociados al uso de servicios comerciales de inteligencia artificial en la nube, calculados principalmente a partir del procesamiento de tokens, el almacenamiento de información, la transferencia de datos y el consumo de interfaces de programación (APIs) ofrecidas por proveedores como OpenAI, Anthropic y Google, a través de modelos comerciales como GPT-5, Claude o Gemini.

3.2.2. Requerimientos de infraestructura local

Las especificaciones técnicas para ejecutar LLMs en servidores institucionales se basaron en documentación oficial de NVIDIA, Meta AI, Hugging Face y reportes de universidades que han implementado LLMs locales (Radas et al., 2024; Dorca Josa and Bleda-Bejar, 2024). La Tabla 3.3 sintetiza estos requerimientos.

Tabla 3.3: Requerimientos técnicos para infraestructura local

Categoría	Requerimiento	Detalles
GPU	Mínimo 24 GB VRAM	Modelos compatibles: NVIDIA RTX 4090, RTX 6000 Ada, A100.
Memoria RAM	32 – 128 GB	Depende del tamaño del modelo y de la concurrencia esperada.
Almacenamiento	SSD NVMe 1 TB	Recomendado para modelos grandes, cuantizados o múltiples instancias.
CPU	8–16 núcleos	Procesadores con soporte AVX/AVX2, como AMD Ryzen 9 o Intel Xeon.
Sistema operativo	Linux (preferido)	Alta compatibilidad con frameworks de IA y servidores REST.
Frameworks	Transformers, vLLM, llama.cpp	Soporte para ejecución optimizada y modelos cuantizados.

Estas especificaciones coinciden con los requerimientos publicados para modelos como LLaMA 3, Mistral y Falcon, cuyos binarios pueden oscilar entre 8 y 140 GB dependiendo de su configuración (Meta AI, 2024; AI, 2024).

3.2.3. Costos estimados de infraestructura local

La Tabla 3.4 presenta los costos estimados para montar un servidor capaz de ejecutar modelos entre 7B y 13B parámetros, que es el rango identificado en §3.1.6 como suficiente para los tres casos de uso de este trabajo. Los rangos en USD se tomaron de catálogos de fabricantes y distribuidores consultados en el primer trimestre de 2026; la conversión a COP se hizo aplicando una tasa de 3.650 COP/USD, valor coherente con la proyección macroeconómica publicada por Fedesarrollo para 2026 (Hernández, 2026). Cada componente se eligió como la opción más asequible que cumple los requerimientos técnicos consolidados en la Tabla 3.3: las notas al pie de la tabla justifican brevemente cada elección.

Tabla 3.4: Costos estimados para infraestructura local (*on-premise*)

Componente	**Especificación**	**Costo (USD)**	**Costo (COP)**
GPU[1]	RTX 4090 / A100 (24–40 GB VRAM)	2.000 – 8.000	7.200.000 – 29.200.000
CPU[2]	Ryzen 9 / Intel Xeon	500 – 1.200	1.825.000 – 4.380.000
RAM[3]	64–128 GB DDR5	300 – 700	1.095.000 – 2.555.000
Almacenamiento[4]	NVMe SSD 2 TB	250 – 400	912.500 – 1.460.000
Energía + mantenimiento anual[5]	Servidor 250 – 650 W	400 – 600	1.460.000 – 2.190.000
Total estimado	—	**3.450 – 10.900**	**12.592.000 – 39.785.000**

Los rangos presentados reflejan variaciones asociadas a disponibilidad de hardware, costos de importación y diferencias de configuración observadas durante el periodo de análisis. Experiencias similares han sido reportadas en instituciones que han explorado despliegues locales de LLMs (Radas et al., 2024; Dorca Josa and Bleda-Bejar, 2024).

Componentes de costo adicionales a considerar

La estimación presentada en la Tabla 3.4 se centra en la inversión en infraestructura y en el consumo energético durante la operación. Sin embargo, existen otros costos que también forman parte del funcionamiento de un servicio basado en LLMs locales y que conviene tener presentes al planificar un despliegue institucional.

- **Reposición del hardware.** Las GPU utilizadas para ejecutar los modelos representan el componente de mayor valor dentro de la infraestructura y, como cualquier equipo sometido a operación continua, eventualmente requieren reemplazo. Considerando los rangos de precio

[1] La RTX 4090 (24 GB) se tomó como referencia de GPU de consumo de alta capacidad disponible durante el periodo de análisis, suficiente para ejecutar modelos cuantizados del rango 7B–14B. La NVIDIA A100 (40–80 GB) representa una categoría de GPU orientada a centros de datos, utilizada como referencia para escenarios que involucren modelos de mayor tamaño, como 70B.

[2] Se consideraron procesadores con soporte para instrucciones AVX-512 (por ejemplo, Ryzen 9 serie 7000 o Xeon Gold), ya que bibliotecas de inferencia como `llama.cpp` y `vLLM` utilizan este tipo de optimizaciones para mejorar el procesamiento asociado a inferencia y carga de modelos.

[3] El rango propuesto cubre configuraciones mínimas y recomendadas según el tamaño del modelo y los servicios auxiliares ejecutados en la misma infraestructura, incluyendo bases vectoriales y componentes de soporte.

[4] Se consideró almacenamiento NVMe debido a que la carga inicial de modelos y el acceso a bases vectoriales pueden verse afectados por latencias de lectura y escritura más altas en otros tipos de almacenamiento.

[5] La estimación incluye el consumo eléctrico continuo del servidor bajo operación permanente (24/7), junto con costos asociados al mantenimiento preventivo y sustitución periódica de componentes de almacenamiento.

utilizados en la Tabla 3.4 (2.000–8.000 USD por GPU), una institución debería contemplar este costo dentro de su planeación financiera de mediano plazo.

- **Administración y operación.** Mantener un servicio de este tipo implica realizar actividades como monitoreo, actualización de software, copias de respaldo y atención de incidentes. En el caso de la Pontificia Universidad Javeriana Cali, estas tareas recaerían sobre la Dirección de Tecnologías de Información. Aunque la Universidad ya dispone de infraestructura para explorar modelos locales, la incorporación de nuevos servicios puede aumentar la dedicación requerida por el equipo responsable. Este trabajo no cuantifica ese costo porque no se contó con información institucional que permitiera estimarlo.

Estos componentes no modifican la comparación presentada en la Tabla 3.4, pero sí amplían los elementos que una institución debería considerar al evaluar el costo de operar modelos de lenguaje sobre infraestructura propia.

3.2.4. Costos basados en la nube

El cálculo de los costos en la nube partió de tres pasos: estimar un volumen mensual representativo para cada caso de uso, tomar las tarifas publicadas en el primer trimestre de 2026 por los tres proveedores comerciales más relevantes (OpenAI, Anthropic y Google), y aplicar las tarifas al volumen para obtener un rango. **Volúmenes mensuales asumidos.** Los volúmenes que aparecen en las tablas siguientes corresponden a un área administrativa de la universidad con un patrón de uso moderado:

- **RAG — 10.000 consultas/mes:** equivalente a unas 333 consultas en día hábil, lo que cabe en el escenario de 30–40 usuarios activos haciendo en promedio 10 consultas diarias cada uno. Se asumió un promedio de 1.500 tokens de entrada (pregunta + contexto recuperado) y 500 tokens de salida por consulta.
- **Transcripción ASR — 500 horas/mes:** equivalente a unas 17 horas diarias de audio, consistente con un escenario donde se transcriben grabaciones de reuniones, sesiones de comités a lo largo del mes.
- **Renombramiento semántico — 5.000 documentos/mes:** equivalente a unos 167 documentos por día hábil, consistente con el flujo del archivo institucional descrito en el Capítulo 5. Se asumió un promedio de 800 tokens de entrada (extracto del documento) y 50 tokens de salida (nombre generado).

Tarifas por proveedor. La Tabla 3.5 resume las tarifas oficiales consultadas en el primer trimestre de 2026 para los modelos representativos de cada proveedor. Se utilizaron los modelos de gama media de cada catálogo, alineados con las tareas de los prototipos.

Tabla 3.5: Tarifas de inferencia por proveedor (USD por millón de tokens, primer trimestre de 2026)

Proveedor	**Modelo**	**Input (USD/MTok)**	**Output (USD/MTok)**	**ASR (USD/min)**
OpenAI	GPT-5	2,50	10,00	0,006
Anthropic	Claude Sonnet 4	3,00	15,00	—
Google	Gemini 2.5 Pro	1,25	5,00	0,024

Anthropic no ofrece servicio propio de transcripción de voz; en un despliegue el componente de reconocimiento automático de voz (ASR) se delegaría en OpenAI Whisper API o en Google Speech-to-Text, de allí que las dos columnas finales sean las únicas opciones disponibles. **Costo total por proveedor.** Aplicando los volúmenes a las tarifas, la Tabla 3.6 muestra el costo mensual y anual desglosado por proveedor.

Tabla 3.6: Costos mensuales y anuales en la nube por proveedor (USD)

Caso de uso	**OpenAI**	**Anthropic**	**Google**	**Rango**
RAG (10.000 consultas)	90	120	44	44 – 120
ASR (500 horas)	180	—	720	180 – 720
Renombramiento (5.000 docs)	12	16	6	6 – 16
Total mensual	282	136 + ASR externo	770	230 – 856
Total anual	3.384	—	9.240	2.760 – 10.272
Total 3 años	10.152	—	27.720	8.280 – 30.816

El rango total a tres años (8.300 – 30.800 USD aproximados) refleja la diferencia entre el escenario más económico (combinar Google Gemini para LLM con Whisper de OpenAI para ASR) y el más costoso (combinar Google para todo, incluyendo Speech-to-Text). Anthropic queda fuera de la columna agregada porque no ofrece ASR. Estos rangos son consistentes con las estimaciones publicadas en análisis comparativos de costos de inferencia (Samsi et al., 2023; Zhang et al., 2024), aunque dichas referencias son de 2023–2024 y los precios actuales son menores; la diferencia ilustra precisamente la advertencia hecha en la introducción de la sección sobre la volatilidad de las tarifas.

3.2.5. Proyección consolidada a uno, dos y tres años

La proyección a varios años permite ver el comportamiento económico de cada modalidad cuando el uso se sostiene en el tiempo. La comparación se hace para los tres casos de uso ya descritos – RAG, transcripción ASR y renombramiento semántico– asumiendo los mismos volúmenes mensuales utilizados en §3.2.4 y manteniendo la configuración de hardware de §3.2.3 para la modalidad local. Para la modalidad local, el CAPEX corresponde al hardware (6.000 USD para una configuración

con GPU profesional de gama media) y el OPEX agrupa consumo eléctrico continuo, mantenimiento y reposición parcial de componentes (400–600 USD/año). Para la modalidad en la nube, el costo crece proporcionalmente al uso; los rangos anuales se toman de la Tabla 3.6.

Tabla 3.7: Comparación de costos totales acumulados a 1, 2 y 3 años (USD)

Modalidad	**CAPEX inicial**	**Año 1**	**Año 2**	**Año 3**
Local	6.000	6.400	7.000	7.600
Nube (mín. – máx.)	0	2.760 – 10.272	5.520 – 20.544	8.280 – 30.816

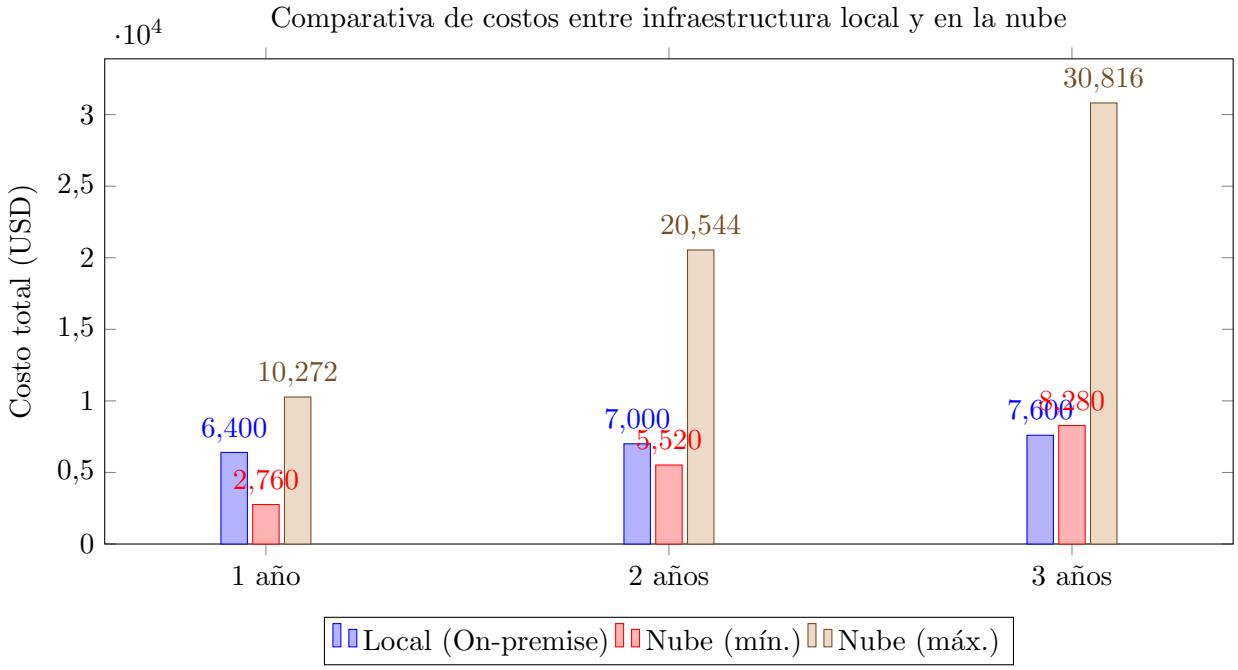

Figura 3.1: Comparación de costos totales estimados (USD) entre ejecución local y en la nube a 1, 2 y 3 años

La comparación evidencia dos estructuras de costo distintas. La modalidad local concentra el gasto en el CAPEX inicial pero mantiene un OPEX bajo y estable, independiente del volumen de uso. La modalidad en la nube elimina la inversión inicial pero genera un costo acumulativo que crece proporcionalmente a la demanda: a partir del año 2 el escenario máximo de la nube ya supera al costo total acumulado de la infraestructura local, y a tres años puede llegar a multiplicarlo por cuatro. Para procesos institucionales con flujos de trabajo continuos y estables, como los tres casos analizados en este proyecto, la ejecución local resulta económicamente más sostenible cuando el uso se mantiene en el tiempo. La nube conserva ventajas para escenarios de uso puntual, experimental

o de demanda altamente variable, lo que abre la posibilidad de un enfoque híbrido en el que cada caso de uso se asigne a la modalidad más conveniente.

La comparación presentada considera los costos estimados de infraestructura, operación y consumo energético, pero no incluye otros componentes descritos en la Sección 3.2.3, como la reposición del hardware ni el tiempo dedicado por el personal de TI a la administración de los servidores. Si estos componentes se incorporan a la estimación, la diferencia económica entre ambas alternativas disminuye y el momento en que la modalidad local empieza a resultar más conveniente puede desplazarse.

No todas las instituciones parten de las mismas condiciones. Una organización que ya dispone de infraestructura y de un equipo encargado de administrarla probablemente asumirá un costo adicional menor que otra que deba crear esa capacidad desde cero. En este último caso, una solución basada en la nube puede seguir siendo una alternativa razonable, especialmente cuando el volumen de uso es moderado. Estimar estos costos con mayor precisión requiere información institucional que estuvo fuera del alcance de este trabajo.

3.3. Implicaciones de privacidad en el uso de LLMs en infraestructura propia

El despliegue de modelos de lenguaje grandes (LLMs) en entornos universitarios *on-premise* ofrece ventajas en cuanto a control y soberanía de los datos; sin embargo, introduce también desafíos en materia de privacidad, seguridad y cumplimiento normativo. En particular, el tratamiento de información académica, administrativa y personal debe garantizarse conforme a las políticas institucionales y las regulaciones nacionales e internacionales de protección de datos, tales como el Reglamento General de Protección de Datos (GDPR) en Europa, la Ley 1581 de 2012 en Colombia y, en contextos específicos como la salud, la HIPAA en Estados Unidos. Esta sección examina los principales riesgos y consideraciones de privacidad asociados al uso de LLMs locales dentro del contexto universitario, y presenta un conjunto de medidas de mitigación orientadas a asegurar la adopción responsable de agentes de inteligencia artificial.

3.3.1. Contexto normativo y ético

La protección de datos personales en entornos educativos es un principio que busca evitar el uso indebido de información sensible, especialmente cuando se emplean tecnologías de inteligencia artificial con capacidad de inferencia y aprendizaje (UNESCO, 2021). Según la **Ley 1581 de 2012**, toda institución debe garantizar la confidencialidad, integridad y disponibilidad de los datos, estableciendo medidas técnicas y administrativas para su resguardo (Congreso de la República de Colombia, 2012). A nivel internacional, el **Reglamento General de Protección de Datos (GDPR)** enfatiza la necesidad del consentimiento informado, la minimización de datos y la transparencia en el procesamiento automatizado (European Union, 2016; Voigt and Von dem Bussche, 2017). Asimismo, la **HIPAA** regula el uso de información médica en contextos de salud y ha sido referenciada en investigaciones sobre IA aplicada a hospitales y universidades (Li, 2023; Dennstädt

et al., 2025). En este sentido, la ejecución de modelos de IA en infraestructura institucional no exime de los principios fundamentales de protección de datos; al contrario, exige una mayor responsabilidad al mantener el control directo sobre el procesamiento.

3.3.2. Riesgos de privacidad en el uso de LLMs locales

La ejecución local LLMs reduce la necesidad de transferir información hacia proveedores externos, pero no elimina los riesgos asociados al tratamiento de datos sensibles. Persisten desafíos relacionados con el control de acceso, la exposición accidental de información, el almacenamiento de datos temporales y las integraciones con otros sistemas.

Metodología de identificación de riesgos. La identificación de riesgos se realizó mediante una revisión cruzada de fuentes técnicas, normativas y de seguridad de la información. Este enfoque buscó evitar un listado construido únicamente desde criterios técnicos o, por el contrario, limitado exclusivamente a exigencias regulatorias. La Tabla 3.8 se elaboró integrando cuatro referencias principales:

- **Taxonomías de seguridad específicas para LLMs:** se utilizó como punto de partida el *OWASP Top 10 for Large Language Model Applications* (OWASP Foundation, 2025), mantenido por la *Open Worldwide Application Security Project Foundation*. Este catálogo identifica vulnerabilidades frecuentes en aplicaciones basadas en LLMs, entre ellas *Sensitive Information Disclosure*, *Improper Output Handling* y *Excessive Agency*. En el contexto de este trabajo se conservaron las categorías con pertinencia para despliegues *on-premise*, dejando por fuera aquellas asociadas principalmente a APIs comerciales en la nube.

- **Literatura académica sobre vulnerabilidades en LLMs:** se revisaron los riesgos documentados por Das et al. (2024) y Zhang et al. (2024), especialmente aquellos relacionados con ataques de *membership inference*, persistencia de tokens en memoria caché y posibles fugas de información derivadas de integraciones externas. Estas referencias permitieron ampliar amenazas que en OWASP aparecen descritas de forma general.

- **Marcos normativos aplicables al contexto institucional:** los riesgos se contrastaron con obligaciones derivadas de la Ley 1581 de 2012 (Congreso de la República de Colombia, 2012) y del GDPR (European Union, 2016), en particular el Artículo 35 relacionado con *Data Protection Impact Assessment*. Esta revisión permitió incorporar escenarios de riesgo con implicaciones jurídicas, como el uso de datos personales en procesos de ajuste fino (*fine-tuning*) sin mecanismos adecuados de anonimización.

- **Estándares de seguridad de la información:** la norma ISO/IEC 27001 (ISO, 2022) y el *AI Risk Management Framework* del NIST (National Institute of Standards and Technology, 2023) se utilizaron para verificar la existencia de controles aplicables a los riesgos identificados, evitando incluir amenazas cuya mitigación no pudiera traducirse en acciones operativas concretas dentro de un entorno institucional.

Después de esta revisión, los riesgos se filtraron con base en dos criterios: su pertinencia dentro de un despliegue local orientado a procesos académico-administrativos y la posibilidad de aplicar medidas de mitigación para el departamento de TI de la universidad. Los riesgos resultantes se presentan en la Tabla 3.8.

Tabla 3.8: Riesgos potenciales de privacidad en LLMs locales

Tipo de riesgo	**Descripción**	**Impacto potencial**
Filtración de datos sensibles	El modelo podría retener o reproducir información privada de estudiantes o administrativos durante la inferencia.	Violación de confidencialidad y reputación institucional.
Acceso no autorizado	Usuarios internos o externos podrían acceder a logs o bases de datos sin los debidos controles de rol.	Exposición de datos personales y académicos.
Persistencia de información en caché	Algunos frameworks (p. ej. *Transformers*, *vLLM*) mantienen memoria temporal de tokens procesados.	Riesgo de recuperación no intencional de datos sensibles.
Entrenamiento o fine-tuning sin anonimización	La inclusión de datos personales sin procesos de seudonimización o anonimización puede derivar en infracciones legales.	Sanciones por incumplimiento de normativas (Ley 1581, GDPR).
Integraciones inseguras	APIs o agentes conectados a servicios externos podrían transmitir datos fuera del entorno local.	Pérdida de control sobre el flujo de información.

3.3.3. Análisis de vulnerabilidades y evaluación de impacto

La identificación de riesgos potenciales necesita complementarse con un análisis de los puntos de la arquitectura donde esos riesgos pueden materializarse y del impacto que tendrían sobre la información institucional. Con este propósito se utilizó la metodología *Privacy Impact Assessment* (PIA), conocida en el contexto del GDPR como *Data Protection Impact Assessment* (DPIA). Este enfoque permite examinar de manera estructurada cómo circulan los datos dentro de un sistema, qué amenazas pueden surgir en cada etapa y qué medidas pueden incorporarse para reducir posibles afectaciones sobre la privacidad de los titulares de la información.

La metodología se encuentra formalizada en la norma *ISO/IEC 29134:2017* (International Organization for Standardization, 2017) y el GDPR (European Union, 2016), a través del Artículo 35, establece su uso en escenarios donde el tratamiento de datos personales involucra tecnologías nuevas y puede implicar riesgos para los derechos de las personas. En el caso de un despliegue institucional de LLMs, esta evaluación adquiere relevancia debido al tratamiento potencial de documentos administrativos, registros académicos o interacciones que pueden contener información sensible.

La aplicación del PIA sobre el flujo descrito en el Capítulo 4 , carga de documentos en la base vectorial, inferencia del modelo a partir del contexto recuperado y almacenamiento de registros de interacción, permitió identificarlas siguientes vulnerabilidades con impacto potencial sobre privacidad y gobernanza de datos:

- Falta de separación entre entornos de desarrollo y producción, con riesgo de exposición de datos utilizados en pruebas.
- Ausencia de políticas de retención y eliminación de información utilizada en procesos de entrenamiento o ajuste fino de modelos.
- Limitaciones en la trazabilidad de accesos y consultas realizadas sobre el sistema.
- Escasa documentación sobre la procedencia y control de los *datasets* empleados.

Estos hallazgos también han sido documentados en estudios sobre vulnerabilidades en LLMs (Das et al., 2024; Zhang et al., 2024), donde se advierte que incluso implementaciones locales pueden presentar fugas de información o vulnerabilidades asociadas a *membership inference attacks*. Este tipo de ataque busca determinar si un dato específico hizo parte del conjunto utilizado para entrenar o ajustar un modelo, aun cuando el atacante no tenga acceso directo a dicho conjunto. En algunos escenarios, el análisis repetido de las respuestas del modelo puede revelar indicios sobre la presencia de información sensible o permitir reconstruir fragmentos parciales del contenido original. El trabajo de Shokri et al. (2017) mostró la viabilidad de estos ataques en modelos de aprendizaje automático tratados como cajas negras, y estudios posteriores han explorado riesgos similares en LLMs ajustados con datos sensibles.

Dentro del contexto de este proyecto, un escenario de riesgo podría surgir si un modelo local fuera ajustado utilizando documentos institucionales con información personal, por ejemplo, expedientes académicos o comunicaciones internas y posteriormente un usuario con acceso al sistema intentara inferir si ciertos datos hicieron parte del proceso de ajuste.

Los prototipos desarrollados en este trabajo no realizan ajuste fino (*fine-tuning*). Todos operan sobre modelos previamente entrenados mediante inferencia y recuperación aumentada por generación (RAG), por lo que el riesgo de *membership inference* descrito anteriormente no aparece en las implementaciones evaluadas.

Aun así, este escenario se incluyó en la matriz de riesgos porque la arquitectura propuesta no depende de un modelo específico. Si en una etapa posterior la institución decide ajustar un modelo con información propia, ese riesgo pasa a ser relevante y debería considerarse dentro de las medidas de privacidad y gobernanza. Por esta razón, se presenta como un escenario posible para futuras implementaciones y no como una vulnerabilidad observada durante el desarrollo de los prototipos.

3.3.4. Recomendaciones para la mitigación de riesgos

La identificación de riesgos de privacidad requiere definir medidas concretas de mitigación que puedan incorporarse dentro de la operación institucional de un sistema basado en LLMs. En el caso de despliegues locales, estas medidas no dependen únicamente de mecanismos técnicos; también involucran controles organizativos y criterios derivados de la normativa sobre protección de datos. Aspectos como el cifrado de información, el aislamiento de entornos o la gestión de actualizaciones deben complementarse con políticas de retención, control de accesos y mecanismos de trazabilidad para reducir la probabilidad de exposición indebida de información sensible.

La Tabla 3.9 reúne las medidas seleccionadas para el contexto de este trabajo y describe su aplicación dentro de un entorno institucional. Cada una incluye una explicación breve de su propósito, la forma en que puede implementarse dentro de un despliegue institucional y el marco normativo o estándar que la respalda. La selección se construyó tomando como referencia obligaciones derivadas de la Ley 1581 de 2012 (Congreso de la República de Colombia, 2012), el GDPR (European Union, 2016), la norma ISO/IEC 27001 (ISO, 2022) y marcos de gobernanza de IA como el *AI Risk Management Framework* del NIST (National Institute of Standards and Technology, 2023) y las recomendaciones de la OCDE (Organisation for Economic Co-operation and Development, 2019).

Tabla 3.9: Medidas recomendadas de mitigación de riesgos basadas en marcos normativos y estándares

Medida	**En qué consiste y cómo se aplica**	**Justificación normativa**
Anonimización y seudonimización	La anonimización busca eliminar o transformar información que permita identificar directamente a una persona, mientras que la seudonimización reemplaza identificadores directos por referencias artificiales separadas de la identidad original. Estas técnicas pueden aplicarse antes de incorporar documentos, transcripciones u otros datos institucionales a sistemas de recuperación o procesamiento basado en LLMs.	Mecanismos contemplados en el GDPR (Art. 4 y 25) y la Ley 1581 de 2012 para reducir riesgos asociados al tratamiento automatizado de datos personales (European Union, 2016; Congreso de la República de Colombia, 2012).
Control de accesos basado en roles (RBAC)	Modelo de autorización que asigna permisos según funciones o responsabilidades institucionales (administrativo, docente, auditor o equipo de TI). Puede utilizarse para restringir acciones como carga de documentos, acceso a información sensible o administración de componentes del sistema.	Práctica recomendada en ISO/IEC 27001 (Anexo A.9, Control de acceso) para limitar el acceso a sistemas y registros sensibles únicamente a personal autorizado (ISO, 2022).

Continúa en la página siguiente

(continuación de la Tabla 3.9)

Medida	En qué consiste y cómo se aplica	Justificación normativa
Cifrado de datos en tránsito y en reposo	Aplicar mecanismos de cifrado para proteger la información durante la transmisión entre componentes del sistema y en el almacenamiento de bases vectoriales, archivos temporales y registros de actividad. Esto incluye protocolos seguros de comunicación y cifrado del contenido almacenado para reducir riesgos de acceso no autorizado.	Recomendado por el *AI Risk Management Framework* del NIST para proteger la confidencialidad e integridad de la información frente a accesos indebidos o interceptación (National Institute of Standards and Technology, 2023).
Auditoría y registro de actividades	Mantener registros centralizados de operaciones relevantes, incluyendo accesos al sistema, consultas realizadas, modificaciones al corpus documental y eventos administrativos. Los registros pueden utilizarse para seguimiento de incidentes, detección de comportamientos anómalos y procesos de auditoría institucional.	Respaldado por ISO/IEC 27001 y marcos de gobernanza de IA responsable de la OCDE para fortalecer trazabilidad, rendición de cuentas y gestión de incidentes (Organisation for Economic Co-operation and Development, 2019; ISO, 2022).
Política de retención de datos	Definir periodos de conservación diferenciados según el tipo de información procesada y establecer procedimientos de eliminación al finalizar el periodo definido. Esta política puede aplicarse a registros operativos, documentos institucionales y datos temporales generados durante la interacción con modelos de lenguaje.	Principio de limitación del almacenamiento del GDPR (Art. 5.1.e) y la Ley 1581 de 2012, que establecen que los datos personales no deben conservarse más tiempo del necesario (European Union, 2016; Congreso de la República de Colombia, 2012).

Continúa en la página siguiente

(continuación de la Tabla 3.9)

Medida	En qué consiste y cómo se aplica	Justificación normativa
Aislamiento de entornos	Separar técnicamente los entornos de desarrollo, pruebas y producción mediante configuraciones independientes de infraestructura, credenciales y datos utilizados. Cuando sea necesario trabajar con información institucional en entornos de prueba, se recomienda anonimizarla previamente.	Práctica recomendada en ISO/IEC 27001 y el NIST para reducir riesgos de exposición de información sensible y limitar la propagación de incidentes entre entornos (ISO, 2022; National Institute of Standards and Technology, 2023).
Actualizaciones y parches de seguridad	Mantener procedimientos periódicos de actualización para sistemas operativos, bibliotecas, modelos y herramientas asociadas al despliegue de LLMs, junto con mecanismos de respuesta ante vulnerabilidades críticas reportadas en componentes del sistema.	Recomendado por estándares de seguridad informática del NIST para mitigar vulnerabilidades conocidas en software y dependencias tecnológicas (National Institute of Standards and Technology, 2023).

3.3.5. Síntesis de riesgos y consideraciones finales

La Tabla 3.10 consolida los hallazgos de las subsecciones anteriores en una matriz que cruza cada riesgo con su consecuencia probable, una estimación cualitativa de probabilidad y la medida principal de mitigación.

Tabla 3.10: Matriz consolidada de riesgos de privacidad y medidas de mitigación

Riesgo	**Consecuencia**	**Probabilidad**	**Medida principal de mitigación**
Filtración de datos	Pérdida de confidencialidad institucional	Media	Anonimización, aislamiento de entornos
Acceso no autorizado	Exposición de registros o credenciales	Alta	RBAC, auditoría y monitoreo
Persistencia de caché	Recuperación no intencional de datos	Media	Limpieza periódica de memoria temporal
Fine-tuning con datos personales	Sanciones por incumplimiento de GDPR/Ley 1581	Alta	Seudonimización y consentimiento informado
Integraciones inseguras	Fuga de información hacia servicios externos	Media	Validación de endpoints, proxies internos

3.3.6. Análisis integrado de hallazgos y conclusiones del capítulo

El análisis desarrollado a lo largo de este capítulo permite identificar patrones comunes y resultados transversales en relación con las implicaciones de privacidad asociadas al uso de modelos de lenguaje grandes (LLMs) ejecutados en infraestructura institucional. En primer lugar, la revisión del contexto normativo evidencia que la ejecución local de modelos de IA no reduce las obligaciones legales en materia de protección de datos; por el contrario, traslada a la institución la responsabilidad directa sobre el cumplimiento de principios como la confidencialidad, la minimización de datos, la transparencia y la rendición de cuentas. Este hallazgo implica que la adopción de LLMs locales debe ir acompañada de mecanismos formales de gobernanza de datos, alineados con regulaciones como la Ley 1581 de 2012, el GDPR y marcos internacionales de ética en inteligencia artificial.

En segundo lugar, el análisis de riesgos muestra que, si bien el despliegue *on-premise* reduce la exposición de datos hacia terceros, persisten riesgos relevantes asociados a la gestión interna de la infraestructura. Los principales riesgos identificados no están vinculados exclusivamente al modelo de lenguaje, sino a los componentes periféricos del sistema, tales como el manejo de logs, la persistencia de información en memoria temporal, los controles de acceso y las integraciones entre servicios. Esto sugiere que la privacidad en entornos de IA depende de las prácticas operativas y de seguridad adoptadas por la institución.

Adicionalmente, la evaluación de vulnerabilidades mediante el enfoque *Privacy Impact Assessment* permitió evidenciar que los riesgos con mayor impacto potencial están asociados a la falta de separación entre entornos, a la ausencia de políticas claras de retención de datos y a una trazabilidad limitada de los accesos a los sistemas. Estos hallazgos indican que, sin controles organizacionales

adecuados, incluso una implementación local puede derivar en escenarios de incumplimiento normativo o exposición indebida de información sensible. Finalmente, las recomendaciones de mitigación propuestas se derivan directamente de los riesgos y vulnerabilidades identificados y se fundamentan en estándares y marcos normativos reconocidos. En conjunto, estas medidas permiten reducir la probabilidad y el impacto de incidentes de privacidad, siempre que se implementen de manera sistemática y se integren en las políticas institucionales existentes.

Como resultado del análisis, se concluye que la adopción de agentes de IA basados en LLMs locales en el contexto universitario es viable desde la perspectiva de privacidad, siempre que la institución asuma una gestión activa de los riesgos, establezca controles técnicos y organizacionales coherentes y adopte un marco de gobernanza de IA que articule la innovación tecnológica con la protección de los derechos digitales de estudiantes y personal académico.

3.4. Síntesis

El análisis realizado permite concluir que la ejecución local de modelos de lenguaje en entornos institucionales es técnicamente viable, siempre que se consideren de manera conjunta los requerimientos de infraestructura, los costos de operación y las medidas de seguridad necesarias para el tratamiento de la información. Asimismo, la adopción puede realizarse mediante diferentes enfoques tecnológicos, que incluyen herramientas con interfaces gráficas para usuarios técnicos, soluciones programáticas orientadas a integración con sistemas institucionales y plataformas de orquestación que permiten construir flujos automatizados. Estos elementos proporcionan al área de tecnología un marco de referencia para la planeación y evaluación de iniciativas de inteligencia artificial basadas en modelos locales.

Capítulo 4

Diseño e implementación de prototipos

4.1. Diseño e implementación de prototipos funcionales de agentes de IA con LLMs locales

Este capítulo presenta el diseño e implementación de tres prototipos de agentes de inteligencia artificial basados en modelos de lenguaje grandes (LLMs) ejecutados localmente. El objetivo es concretar los requerimientos técnicos, económicos y de seguridad analizados en el capítulo anterior en escenarios prácticos de uso dentro de un contexto organizacional.

4.1.1. Contexto institucional y supuestos de implementación

Los supuestos utilizados para el diseño de los prototipos se definieron a partir de la revisión de la literatura y de 5 sesiones de trabajo realizadas con la Dirección de Tecnologías de Información (DTI) de la Pontificia Universidad Javeriana Cali durante el desarrollo del proyecto. En estas reuniones se discutieron los resultados obtenidos con los prototipos, las condiciones actuales de la infraestructura institucional y los aspectos que podrían influir en una eventual adopción de este tipo de tecnologías.

Al inicio del trabajo se consideró al Archivo Central como el principal actor institucional, debido a su relación con algunos de los procesos analizados. Sin embargo, conforme avanzó la investigación fue tomando mayor relevancia el papel de la Dirección de Tecnologías de Información. Las decisiones relacionadas con infraestructura, administración de servicios, seguridad de la información y despliegue de modelos de lenguaje recaen sobre esta dependencia, por lo que el análisis terminó orientándose desde esa perspectiva.

A partir de estas discusiones, los prototipos se diseñaron considerando un entorno organizacional en el que existe un área de tecnologías de la información responsable de la infraestructura, la administración de los servicios y el control de accesos a los sistemas institucionales. Bajo este contexto se asumieron los siguientes supuestos:

- La infraestructura de cómputo (servidores, máquinas virtuales o servicios institucionales) es administrada por el área de TI.
- Los modelos de lenguaje y los servicios asociados se despliegan de manera centralizada.

- Los usuarios interactúan con los sistemas mediante interfaces web, carpetas compartidas o flujos automatizados, sin realizar configuraciones técnicas en sus equipos.
- El procesamiento de la información se ajusta a las políticas institucionales de seguridad y privacidad.

Estos supuestos sirvieron como marco para el diseño y la evaluación de los tres prototipos desarrollados en este trabajo. Aunque cada uno responde a un caso de uso diferente, todos fueron planteados pensando en un despliegue administrado desde la infraestructura institucional y en un escenario donde el procesamiento de la información permanece bajo control de la Universidad.

4.2. Selección y justificación de los casos de uso

Los tres casos de uso seleccionados recuperación documental con RAG, transcripción automática y renombramiento semántico comparten tres rasgos que los hacían pertinentes para este trabajo: son procesos de alto volumen y tarea repetitiva en una universidad, son tareas en las que un LLM aporta valor medible, todos manejan información que la institución no puede ni quiere enviar a un servicio comercial en la nube. Esta sección presenta solo la motivación funcional de cada caso; la descripción técnica de cada prototipo se desarrolla en las secciones siguientes.

Caso 1: recuperación documental con RAG

La necesidad de consultar documentación interna en lenguaje natural es transversal a varias áreas de una universidad: el equipo de soporte de TI consulta manuales y procedimientos para resolver tickets, recursos humanos consulta manuales internos y políticas para responder preguntas de personal, registro académico atiende preguntas frecuentes sobre normativa de matrícula y reglamentos. En todos estos escenarios la información existe pero está distribuida en múltiples repositorios (carpetas compartidas, intranet, correos, archivos PDF), lo que hace que parte importante del tiempo de atención se gaste en localizar el documento correcto en lugar de responder la pregunta. Un agente de tipo RAG aborda directamente este problema al indexar los documentos institucionales y permitir consultas en lenguaje natural sobre ellos.

La razón para resolver este caso de uso con un LLM local y no con un servicio en la nube es que parte del corpus relevante incluye información que no puede subirse a servidores externos: procesos disciplinarios, comunicaciones internas con datos personales, expedientes académicos sensibles. Al limitar la consulta a un modelo desplegado dentro del perímetro institucional, los documentos no abandonan en ningún momento la infraestructura de la universidad. Esta es la condición que distingue este caso de uso de un chatbot genérico construido sobre la API de OpenAI o Anthropic.

Caso 2: transcripción automática de reuniones y clases

La transcripción de reuniones y clases continúa siendo una tarea frecuente dentro de los procesos académico-administrativos universitarios. Comités, sesiones de seguimiento, reuniones virtuales

o clases grabadas suelen requerir la elaboración posterior de actas, resúmenes o registros de seguimiento, una labor que normalmente implica escuchar nuevamente el contenido y organizar la información relevante.

Diversas plataformas de videoconferencia ofrecen funciones integradas de transcripción, entre ellas Microsoft Teams, Google Meet y Zoom. Sin embargo, en contextos institucionales pueden surgir limitaciones relacionadas tanto con el tratamiento de la información como con el desempeño del reconocimiento de voz en escenarios específicos. En reuniones donde aparecen términos técnicos, nombres propios o vocabulario especializado de determinadas disciplinas, la precisión de la transcripción puede verse afectada. Además, el procesamiento del audio suele depender de infraestructura administrada por el proveedor, lo que puede generar restricciones cuando las conversaciones involucran información sensible, como procesos disciplinarios, casos académicos individuales, información laboral o deliberaciones internas.

Una alternativa consiste en utilizar modelos de reconocimiento de voz (ASR) ejecutados localmente, como Whisper. De esta manera, el procesamiento del audio permanece dentro de la infraestructura institucional. Este enfoque también permite adaptar el flujo de trabajo a necesidades particulares, por ejemplo, la generación automática de borradores de actas o la integración con sistemas internos de almacenamiento documental.

Caso 3: renombramiento semántico de documentos

La gestión documental forma parte de múltiples procesos académico-administrativos dentro de una universidad. Informes, oficios, certificados, contratos, actas y comunicaciones se generan y almacenan diariamente en formato digital, lo que hace necesario mantener criterios de organización que faciliten su localización posterior.

En la práctica, los nombres asignados a los archivos suelen variar entre dependencias o personas responsables del registro. Esto puede dificultar la recuperación posterior de documentos, especialmente cuando existen diferencias en abreviaturas, formatos de fecha o descripciones utilizadas para nombrar archivos similares. También incrementa el trabajo asociado a tareas de organización documental y verificación previa al archivo institucional.

El uso de agentes basados en LLMs abre la posibilidad de apoyar este proceso mediante el análisis del contenido de un documento para sugerir nombres consistentes con una convención institucional previamente definida. De esta manera, el sistema puede contribuir a mantener criterios homogéneos de nomenclatura y facilitar procesos posteriores de búsqueda, clasificación y auditoría documental.

Rasgo común a los tres casos

Los tres casos comparten el hecho de que parte del corpus que procesan contiene datos institucionales o personales sensibles, y que la calidad funcional del servicio depende de poder operar sobre la totalidad de ese corpus, no solo sobre un subconjunto público. Esto justifica que se aborden los tres con LLMs locales en lugar de servicios comerciales, y es la razón por la que aparecen agrupados como objeto de este capítulo.

Criterios de fondo para la selección

La selección de los tres casos de uso también estuvo condicionada por las circunstancias en las que se desarrolló el proyecto. Aunque existen otros procesos de la universidad donde los agentes de IA podrían aportar valor, como el apoyo a los procesos de acreditación, la clasificación automática de correos o las homologaciones, abordarlos habría requerido condiciones diferentes a las disponibles durante esta investigación.

- **Etapa de exploración institucional.** Durante el desarrollo del proyecto, la Universidad adelantaba las primeras iniciativas relacionadas con el uso de LLMs locales. Por esta razón, se priorizaron casos de uso que pudieran implementarse, evaluarse y reproducirse con la infraestructura disponible.
- **Disponibilidad de información.** Todas las pruebas se realizaron con documentos de acceso público. El uso de información institucional confidencial requería un proceso de autorización que excedía el tiempo previsto para este trabajo.
- **Tiempo de desarrollo.** El proyecto se ejecutó en aproximadamente cuatro meses y medio, lo que hizo necesario concentrar el esfuerzo en tres prototipos y un conjunto acotado de pruebas.

Procesos como los de acreditación o las homologaciones requieren un conocimiento más profundo de la operación institucional, además de acceso a información y actores que estuvieron fuera del alcance de esta investigación. Por ese motivo no hicieron parte de los prototipos desarrollados.

4.3. Arquitectura general

Los tres prototipos se diseñaron bajo una arquitectura común basada en los siguientes componentes:

- Modelos de lenguaje ejecutados localmente mediante **Ollama**.
- Contenedores **Docker** para el despliegue y aislamiento de servicios.
- Comunicación entre componentes mediante protocolos HTTP o acceso a directorios compartidos.
- Procesamiento centralizado en el backend, administrado por el área de TI.

Este enfoque permite desacoplar la lógica de inteligencia artificial de las interfaces de usuario, facilitando su integración posterior con aplicaciones institucionales o portales web.

4.3.1. Sobre Ollama y por qué se eligió

Ollama (Ollama, 2025) es el *runtime* de inferencia descrito en §3.1.5: empaqueta modelos cuantizados en formato GGUF junto con un servidor HTTP local y expone una API compatible con la de OpenAI, lo que permite que cualquier aplicación que ya consume modelos comerciales (Open WebUI, n8n, AutoGen, scripts en Python con `requests`) pueda apuntar al servidor local sin cambios de código. La elección entre los cuatro frameworks comparados en §3.1.5 (Ollama, llama.cpp, vLLM, Transformers de Hugging Face) se hizo sobre tres criterios:

- **Tiempo de puesta en marcha:** con Ollama se puede descargar y desplegar un modelo es un solo comando (`ollama pull mistral`; `ollama serve`), mientras que con vLLM o Transformers el equipo debe escribir el servidor de inferencia, manejar la cuantización y configurar el batching.
- **Compatibilidad de API:** las herramientas integradas en los prototipos (Open WebUI, n8n, AutoGen) consumen la API de OpenAI por defecto. Ollama expone esa misma API en `localhost:11434/v1` sin configuración adicional, lo que evitó escribir adaptadores.
- **Capacidad suficiente para los volúmenes del trabajo:** en §3.1.6 se concluyó que el rango de modelos viable para los tres casos de uso es 7B–14B parámetros. Ollama cubre ese rango sin que las optimizaciones avanzadas de vLLM (PagedAttention, tensor parallelism) sean necesarias.

Modelos disponibles en Ollama relevantes para este trabajo. El catálogo de Ollama incluye decenas de modelos. La Tabla 4.1 resume los que se evaluaron como candidatos para los prototipos.

Tabla 4.1: Modelos relevantes del catálogo de Ollama y su idoneidad por tarea

Modelo	**Tamaño / cuant.**	**Para qué tarea es mejor**
`mistral:7b`	7B / Q4	Generación general en español; equilibrio calidad/velocidad. Es el modelo base usado en RAG y en el postprocesamiento de transcripciones.
`llama3.1:8b`	8B / Q4	Alternativa a Mistral con mejor desempeño en razonamiento de varios pasos; útil para el agente de renombramiento.
`qwen2.5:7b`	7B / Q4	Buen soporte multilingüe (incluye chino y japonés); relevante si la institución necesita procesar documentos en varios idiomas.
`phi-4:14b`	14B / Q4	Razonamiento en tareas estructuradas; opción más capaz cuando hay margen de hardware (24 GB VRAM).
`gemma:7b`	7B / Q4	Modelo abierto de Google; respuesta más concisa, menor consumo de tokens en RAG.
`nomic-embed-text`	137M / FP16	Modelo de *embeddings* usado por Qdrant para indexar el corpus documental en el agente RAG. No genera texto, solo vectores.

Para los prototipos de este trabajo se utilizó `mistral:7b` como modelo de generación principal y `nomic-embed-text` para los *embeddings* del agente RAG. Whisper se ejecuta como un servicio aparte (no a través de Ollama) porque no es un LLM sino un modelo de reconocimiento de voz.

4.4. Enfoque de diseño e implementación

Para el cumplimiento de este objetivo se adoptó un enfoque modular sobre la arquitectura común descrita en §4.3, en el que cada agente se implementó como un prototipo independiente con su propia configuración de Docker Compose, su propia documentación y sus propias pruebas, pero compartiendo Ollama como motor de LLM.

Los prototipos se diseñaron bajo los siguientes principios:

- Ejecución local y sin dependencia de servicios externos.
- Arquitectura modular y reutilizable.
- Registro de resultados y trazabilidad.

4.4.1. Metodología de desarrollo

El desarrollo se realizó siguiendo un ciclo iterativo e incremental basado en los principios de Scrum, lo que permitió ajustar progresivamente cada prototipo, documentar avances y validar resultados parciales en cada iteración. Cada caso de uso se abordó como un microproyecto independiente, con sus propios entregables, pruebas y documentación, pero compartiendo la base tecnológica común (Ollama + Docker Compose).

Etapas de desarrollo

1. **Análisis del proceso y definición del agente.** Se analizaron los procesos académico-administrativos seleccionados, identificando las tareas susceptibles de automatización mediante agentes de IA. Para cada caso, se definieron las entradas, salidas, actores y reglas de negocio, además de los roles y objetivos funcionales de los agentes (analizador, transcriptor, renombrador). Finalmente se seleccionaron los modelos de lenguaje apropiados según la complejidad del proceso y los recursos de cómputo disponibles (Mistral, Whisper, etc.).

2. **Diseño de la arquitectura técnica.** Se especificaron las arquitecturas locales de los tres prototipos, definiendo los contenedores, redes, volúmenes y dependencias mediante Docker Compose. Asimismo, se integraron herramientas complementarias de acuerdo con cada caso de uso:

   - Open WebUI + Qdrant para el agente RAG (caso de uso 1).
   - n8n + Whisper para el agente de transcripción (caso de uso 2).
   - AutoGen Studio para el sistema multiagente de renombramiento (caso de uso 3).

   Durante esta etapa, se diseñaron los flujos de comunicación entre agentes, APIs y bases de datos vectoriales, garantizando un intercambio de información local.

3. **Implementación de los prototipos.** Se implementaron los agentes utilizando Python 3.11, junto con librerías específicas como `autogenstudio`, `requests`, `PyPDF2`, `fastapi`, `whisper` y `langchain`. Cada prototipo fue empaquetado en contenedores Docker, con sus respectivos archivos `Dockerfile` y scripts de inicialización (`main.py`). Además, se habilitaron mecanismos de registro de actividad para facilitar la trazabilidad y evaluación experimental. Todo el código desarrollado fue documentado y versionado en GitHub.

4. **Pruebas funcionales y validación.** Cada agente fue sometido a pruebas controladas que permitieron medir su desempeño y estabilidad: en RAG se evaluó la precisión en la recuperación semántica y la relevancia de las respuestas; en transcripción se midió la fidelidad de los textos obtenidos, la compatibilidad de formatos y el tiempo de procesamiento; en renombramiento se verificó la coherencia y unicidad de los nombres generados a partir del contenido. Se registraron métricas de uso de CPU y RAM, tiempos de respuesta, tamaño de modelos y errores por ejecución para comparar la eficiencia de los tres sistemas.

5. **Evaluación comparativa y documentación.** Finalmente se realizó un análisis comparativo entre los tres prototipos, identificando ventajas, limitaciones y posibles mejoras de cada implementación. Los resultados experimentales fueron documentados y sistematizados, lo que dio lugar a las conclusiones parciales y recomendaciones técnicas que aparecen en el Capítulo 6.

A partir de §4.5 se presenta cada caso de uso por separado siguiendo la misma estructura: necesidad funcional, interacción del usuario, responsabilidades del área de TI, despliegue y descripción técnica del prototipo.

4.5. Caso de uso 1: Recuperación aumentada por generación (RAG)

4.5.1. Necesidad que soporta

Este caso de uso permite realizar consultas en lenguaje natural sobre documentos institucionales, facilitando la localización de información académica o administrativa distribuida entre múltiples fuentes.

4.5.2. Interacción del usuario

El usuario final accede al sistema mediante una interfaz web (Open WebUI), desde la cual puede formular preguntas en lenguaje natural y recibir respuestas generadas por el modelo junto con el contexto recuperado.

4.5.3. Responsabilidades del área de TI

El área de tecnología debe encargarse de:

- Desplegar los contenedores Docker.
- Cargar y actualizar los documentos institucionales.
- Administrar la base vectorial.
- Monitorear el uso y el rendimiento del servicio.

4.5.4. Despliegue de la solución

El agente RAG combina tres componentes que ya fueron presentados en otras secciones del documento: la técnica de **Retrieval-Augmented Generation** descrita en §2.1, que recupera fragmentos del corpus institucional y los entrega como contexto al LLM para anclar las respuestas en documentos auditables y reducir el riesgo de alucinaciones; **Open WebUI** (Open WebUI, 2025), la interfaz web tipo chat que se expone a los usuarios dentro de la red institucional y se conecta a Ollama por debajo; y **Qdrant** (Qdrant, 2025), la base de datos vectorial donde se almacenan los

embeddings del corpus generados con el modelo `nomic-embed-text` y sobre la cual se ejecuta la búsqueda por similitud semántica.

Desde el punto de vista de infraestructura, el agente RAG se implementa como un conjunto de servicios desplegados en un servidor institucional administrado por el área de TI. La solución se ejecuta mediante contenedores Docker que alojan los tres componentes anteriores. El acceso de los usuarios se realiza a través de un navegador web dentro de la red institucional; el procesamiento de las consultas, la recuperación de información y la generación de respuestas se ejecutan completamente en el backend. Este enfoque permite mantener el control sobre los documentos institucionales y facilita la administración centralizada del servicio. La Figura 4.1 muestra el diagrama de despliegue del prototipo.

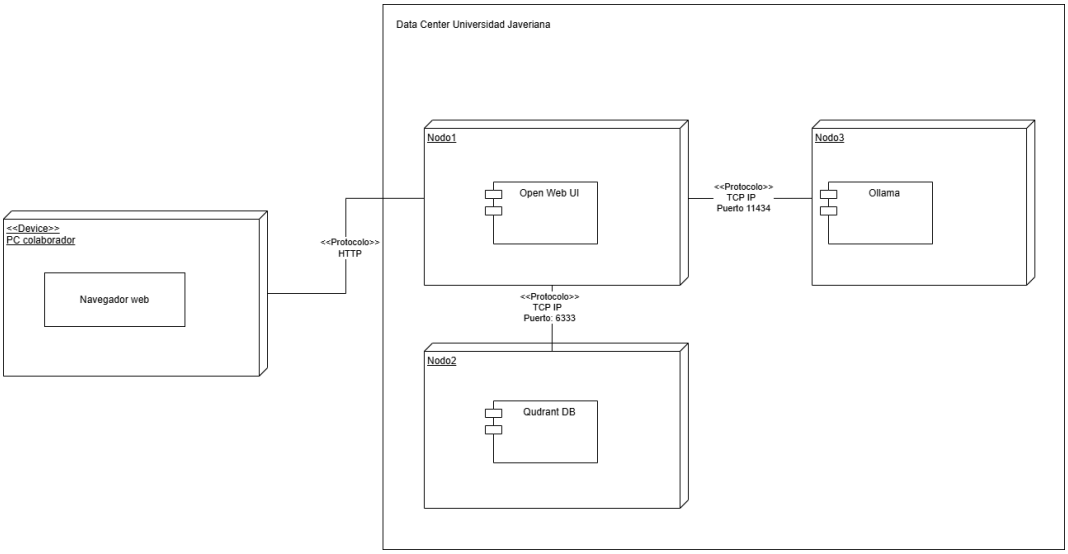

Figura 4.1: Diagrama de despliegue del agente RAG. Los tres nodos pueden corresponder a una sola máquina (despliegue de prototipo, como en este trabajo, donde los tres contenedores corren en el mismo equipo de pruebas descrito en la Tabla 5.1 del Capítulo 5) o a tres máquinas distintas en una arquitectura productiva, comunicadas por la red institucional. La elección depende del volumen de usuarios concurrentes esperado: para los rangos discutidos en §3.1.4, una sola máquina con 16–24 GB de VRAM es suficiente.

4.5.5. Descripción del prototipo

El primer prototipo integró Ollama, Open WebUI y Qdrant para permitir la recuperación semántica y generación contextualizada de respuestas sobre documentos institucionales. El sistema combinó búsqueda vectorial con razonamiento del modelo LLM, sin depender de la nube. La arquitectura implementada se muestra en la Figura 4.2.

Objetivo funcional: Ofrecer respuestas contextualizadas sobre programas académicos y políticas universitarias.

Arquitectura: Docker Compose con tres servicios (ollama, openwebui y qdrant).

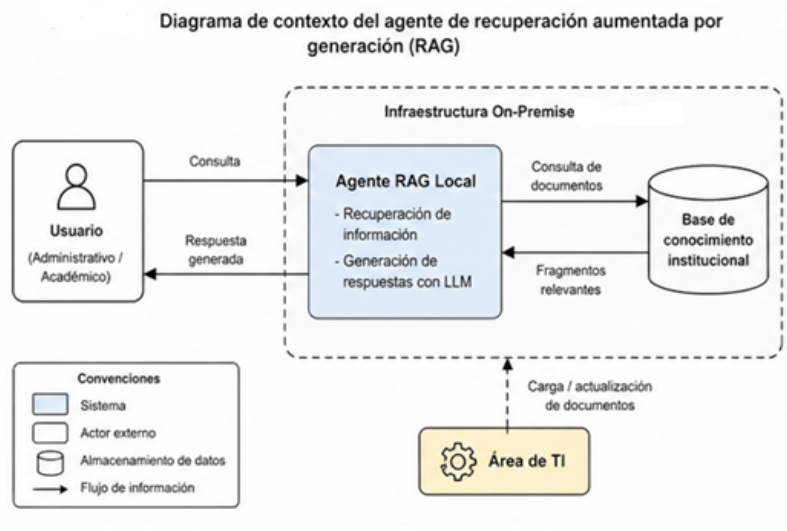

Figura 4.2: Agente RAG — Recuperación Aumentada por Generación

Modelo base: Mistral.
Resultado: Un entorno RAG completamente local, capaz de responder consultas semánticas con privacidad garantizada.

4.6. Caso de uso 2: Transcripción automática

4.6.1. Necesidad que soporta

Este caso de uso permite convertir grabaciones de reuniones, clases o eventos institucionales en texto, facilitando la elaboración de actas y el almacenamiento de información.

4.6.2. Interacción del usuario

El usuario carga el archivo multimedia o lo deposita en una carpeta de entrada, tras lo cual el sistema ejecuta automáticamente el flujo de procesamiento y genera la transcripción y el acta de reunión.

4.6.3. Responsabilidades del área de TI

El área de tecnología debe encargarse de:

- Desplegar los contenedores Docker.

- Configuración del workflow en n8n.
- Gestión de almacenamiento.
- Monitorear el uso y el rendimiento del servicio.

4.6.4. Despliegue de la solución

El agente de transcripción se despliega como un flujo automatizado ejecutado en un servidor institucional. La solución integra un orquestador de procesos (n8n), un servicio de transcripción basado en Whisper y un modelo LLM local para el postprocesamiento del texto.

Los usuarios interactúan con el sistema mediante la carga de archivos en una interfaz web o en un directorio compartido. Una vez recibido el archivo, el flujo de trabajo ejecuta de forma automática las etapas de transcripción, procesamiento y almacenamiento del resultado.

Todos los componentes se ejecutan en contenedores Docker y operan dentro de la infraestructura institucional, lo que permite el control del almacenamiento de archivos multimedia y de las transcripciones generadas.

La Figura 4.3 presenta el diagrama de despliegue del prototipo.

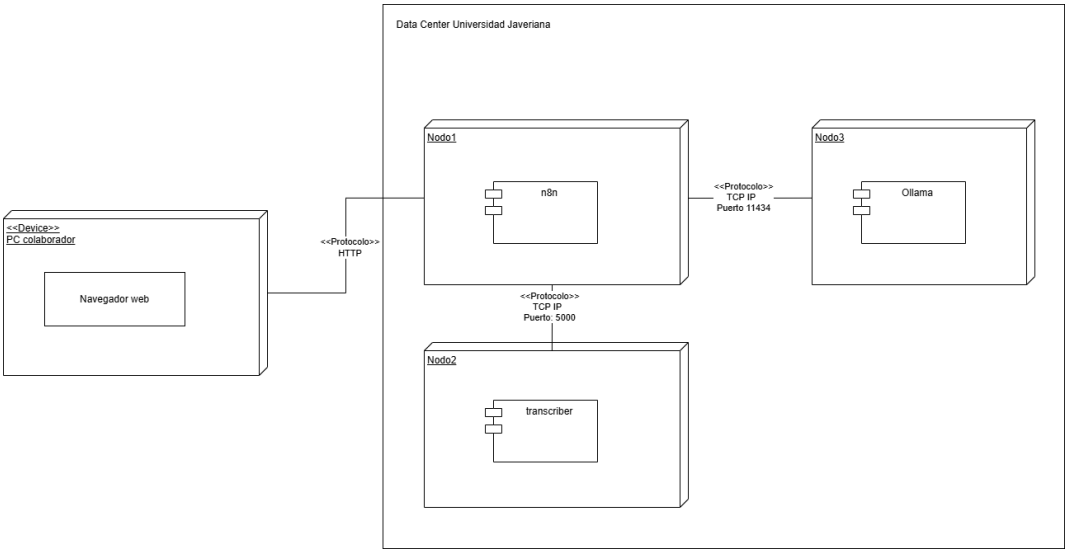

Figura 4.3: Diagrama de despliegue del agente de transcripción automática

4.6.5. Descripción del prototipo

El segundo prototipo automatizó la transcripción de archivos de audio y video mediante la integración de n8n como orquestador de flujos y Whisper como modelo de reconocimiento automático de voz (ASR). Ollama se utilizó como módulo de postprocesamiento para organizar y resumir los textos generados. La arquitectura general del sistema se presenta en la Figura 4.4

Objetivo funcional: Convertir archivos multimedia (MP3, MP4, WAV) en transcripciones textuales estructuradas.

Arquitectura: Microservicio Python transcriber con n8n y Ollama en contenedores Docker.

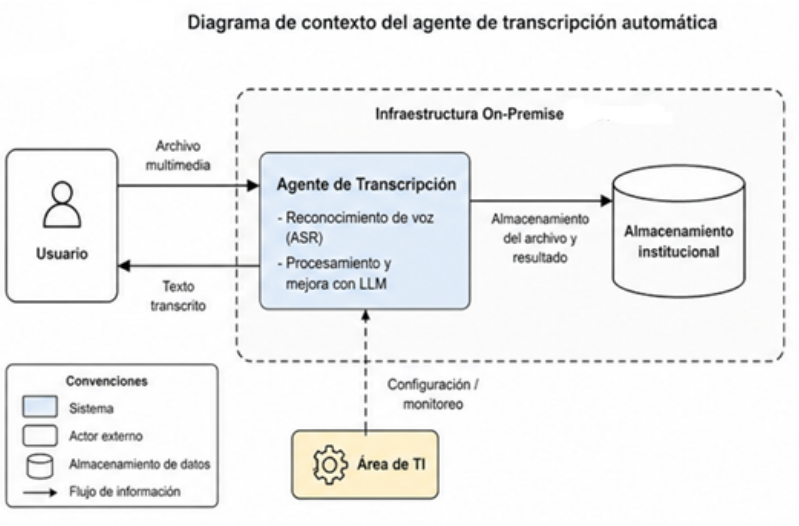

Figura 4.4: Arquitectura del prototipo de transcripción de archivos multimedia con n8n, servicio de transcripción y Ollama.

Resultado: Un flujo reproducible y completamente automatizado para generar transcripciones locales útiles en procesos académicos y administrativos.

4.7. Caso de uso 3: Renombramiento semántico de documentos

4.7.1. Necesidad que soporta

Este caso de uso permite que un agente analice automáticamente cada documento depositado en una carpeta de entrada y le asigne un nombre estandarizado a partir del contenido, eliminando el trabajo manual de nomenclatura y produciendo nombres consistentes en todo el corpus.

4.7.2. Interacción del usuario

El usuario pone uno o varios archivos en la carpeta de entrada del sistema. El servicio detecta automáticamente cada archivo nuevo, lo analiza y deposita el archivo renombrado en la carpeta de salida. No requiere abrir aplicación gráfica ni completar formularios.

4.7.3. Responsabilidades del área de TI

El área de tecnología debe encargarse de:

- Desplegar los contenedores Docker.
- Definir las reglas de nomenclatura institucional que el agente debe seguir.
- Configurar las carpetas de entrada y salida y sus políticas de respaldo.
- Monitorear el uso y el rendimiento del servicio.

4.7.4. Despliegue de la solución

El tercer prototipo introdujo un sistema colaborativo entre agentes utilizando **AutoGen Studio** y Ollama. AutoGen ya fue presentado en §4.3 como el framework de Microsoft Research para construir aplicaciones multiagente sobre LLMs; su elección para este caso de uso responde a que la tarea de renombrar un documento se descompone naturalmente en dos pasos: comprender el contenido y producir un nombre estructurado.

El diseño incluyó un **Agente Analizador**, encargado de leer el documento y extraer su tipo, emisor/destinatario y referencia temática principal, y un **Agente Renombrador**, responsable de combinar esos elementos en un nombre coherente siguiendo el formato de nomenclatura definido por la institución. Ambos agentes se comunican mediante un *GroupChat* de AutoGen, simulando una interacción cooperativa: el Renombrador puede pedir al Analizador que confirme un dato si el extracto inicial es ambiguo, antes de comprometer un nombre final.

El servicio se implementa como un componente backend en contenedores Docker dentro de la infraestructura institucional. Su funcionamiento parte de un directorio de entrada donde se depositan los documentos a procesar; el sistema analiza cada archivo mediante el flujo de agentes descrito y almacena la versión renombrada en un directorio de salida, lo que facilita su incorporación a esquemas ya existentes de gestión documental sin modificar los flujos de trabajo. La Figura 4.5 presenta el diagrama de despliegue correspondiente.

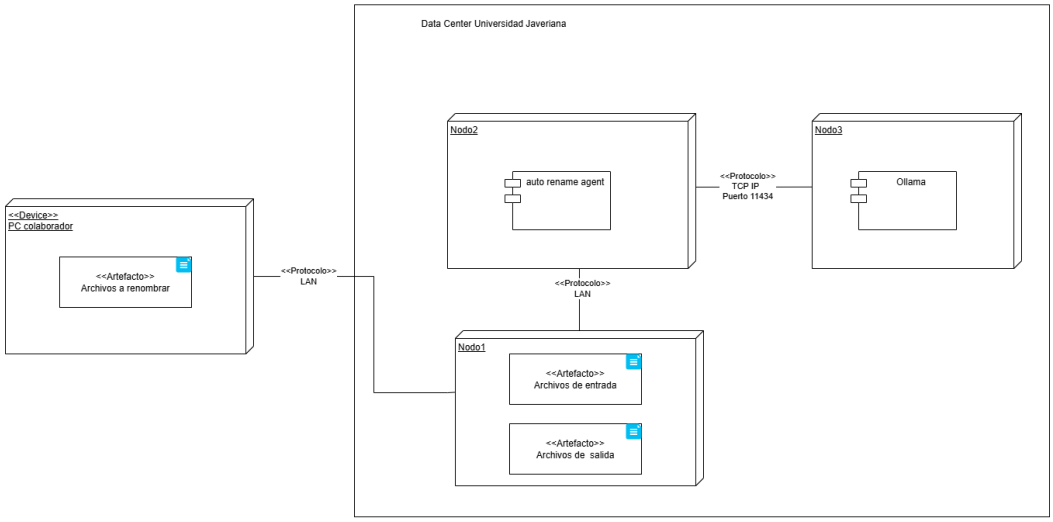

Figura 4.5: Diagrama de despliegue del agente de renombramiento semántico. En el prototipo, los nodos del agente y el motor Ollama corren en la misma máquina; en un despliegue productivo pueden separarse para que un único servidor de inferencia (Ollama) atienda a varios servicios de la institución, incluyendo este agente y el RAG del caso de uso 1.

4.7.5. Descripción del prototipo

El tercer prototipo integró AutoGen Studio y Ollama para implementar un sistema multiagente capaz de analizar el contenido de un documento y producir un nombre estructurado a partir de él. El flujo se apoya en dos agentes especializados que se comunican entre sí mediante un *GroupChat*, lo que permite una colaboración progresiva sin necesidad de intervención del usuario una vez iniciado el procesamiento. La arquitectura implementada se muestra en la Figura 4.6.

Objetivo funcional: analizar y renombrar documentos de forma autónoma y semánticamente coherente.

Arquitectura: servicio de Python con Docker Compose y carpetas `/data/input` y `/data/output` como puntos de entrada y salida del flujo.

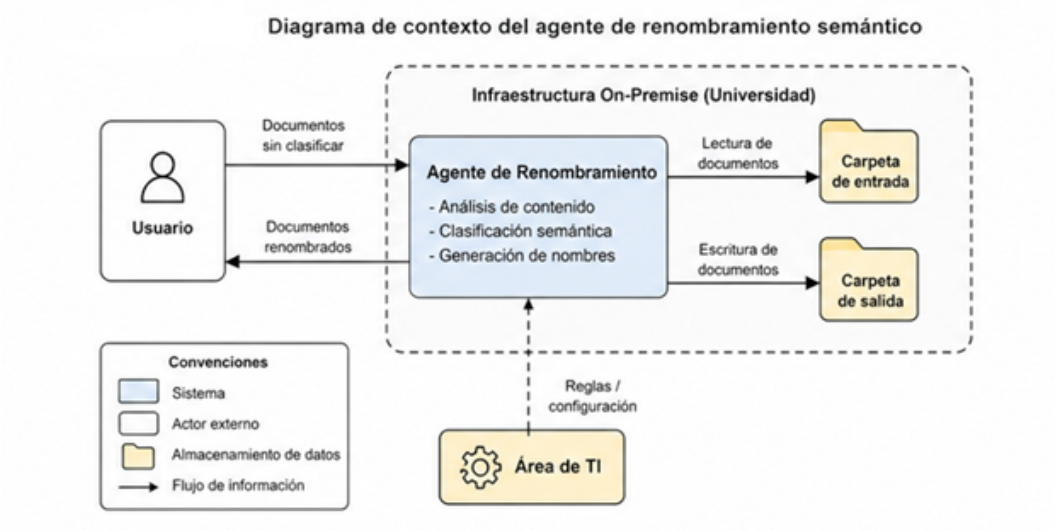

Figura 4.6: Arquitectura del prototipo multiagente de renombramiento automatizado con AutoGen, Ollama y directorios locales de entrada y salida.

Modelo base: `mistral:7b` ejecutado en Ollama, invocado por ambos agentes (Analizador y Renombrador) con *prompts* de sistema diferentes.

Resultado: un sistema multiagente funcional que organizó archivos institucionales de manera automática y trazable.

4.8. Herramientas, entregables y código fuente

4.8.1. Herramientas utilizadas

Las herramientas y tecnologías seleccionadas para el desarrollo de los tres prototipos se alinearon con el objetivo de garantizar ejecución local, modularidad y facilidad de integración entre componentes. En la Tabla 4.2 se presenta un resumen de las principales soluciones utilizadas, su clasificación y el propósito específico que cumplieron dentro de la arquitectura de cada caso de uso.

Tabla 4.2: Herramientas y tecnologías utilizadas en el desarrollo de los prototipos

Tipo	**Herramienta / Tecnología**	**Propósito**
LLM local	Ollama	Ejecución local de modelos de lenguaje grandes (LLMs) como Mistral y soporte a los agentes en los tres casos de uso.
Orquestación de contenedores	Docker Compose	Despliegue, aislamiento y gestión de los servicios involucrados en cada prototipo (LLMs, servicios web, bases de datos, orquestadores).
Recuperación semántica (RAG)	Qdrant	Almacenamiento y consulta de vectores para la recuperación aumentada por generación en el prototipo RAG.
Interfaz conversacional	Open WebUI	Interfaz web para la interacción con el agente RAG y prueba de capacidades de consulta sobre documentos institucionales.
Orquestación de flujos	n8n	Automatización del flujo de transcripción de archivos multimedia, integrando carga de archivos, llamado al servicio de transcripción y manejo de salidas.
ASR (Transcripción)	Whisper / Servicio de transcripción	Modelo de reconocimiento automático de voz para generar transcripciones de audio y video en el prototipo de transcripción.
Framework multiagente	AutoGen / AutoGen Studio	Definición y coordinación de agentes (analizador y renombrador) en el prototipo de renombramiento automatizado.
Lenguaje de programación	Python 3.11	Implementación de la lógica de los agentes, servicios de integración y scripts de ejecución.
Procesamiento de documentos	PyPDF2, herramientas de manejo de texto	Extracción de contenido de documentos de entrada para análisis semántico y generación de nombres de archivo.
Control de versiones y documentación	Git, GitHub, README.md	Gestión del código fuente, trazabilidad de cambios y documentación técnica de cada prototipo.

4.8.2. Entregables

Esta subsección presenta los entregables tangibles producidos por el proyecto: el código fuente de los tres prototipos con sus respectivos archivos de despliegue (`docker-compose.yml`), Dockerfiles

y scripts de inicialización, la documentación técnica en formato `README.md` para reproducir cada despliegue, y los resultados experimentales detallados en el Capítulo 5. Los tres repositorios se publicaron en GitHub bajo licencia **MIT**, lo que permite a la institución reutilizarlos, modificarlos e integrarlos con sus propios sistemas sin restricciones más allá de mantener el aviso de copyright original.

- **Prototipo RAG local:** entorno funcional con Ollama, Qdrant y Open WebUI.
  Repositorio: `https://github.com/edgaralvarezrengifo/llm-stack-local`
- **Agente de transcripción:** flujo completo con n8n y Whisper orquestados mediante Ollama.
  Repositorio: `https://github.com/edgaralvarezrengifo/agent-transcripcion-local`
- **Agente multiagente de renombramiento:** sistema AutoGen–Ollama para clasificación y renombramiento local.
  Repositorio: `https://github.com/edgaralvarezrengifo/llm-file-rename-agent`
- **Documentación técnica:** archivos `README.md` con instrucciones paso a paso de despliegue y uso para cada prototipo.
- **Resultados experimentales:** métricas de desempeño, registros de ejecución y análisis comparativo (Capítulo 5).

4.8.3. Capturas de las soluciones en uso

Los tres prototipos son servicios de backend que un usuario final consume a través de tres interfaces distintas: el chat web de Open WebUI (caso de uso 1), el editor de flujos de n8n (caso de uso 2) y el directorio de entrada/salida del agente de renombramiento (caso de uso 3). Las Figuras 4.7, 4.8 y 4.9 ilustran el aspecto que ve el usuario en cada caso.

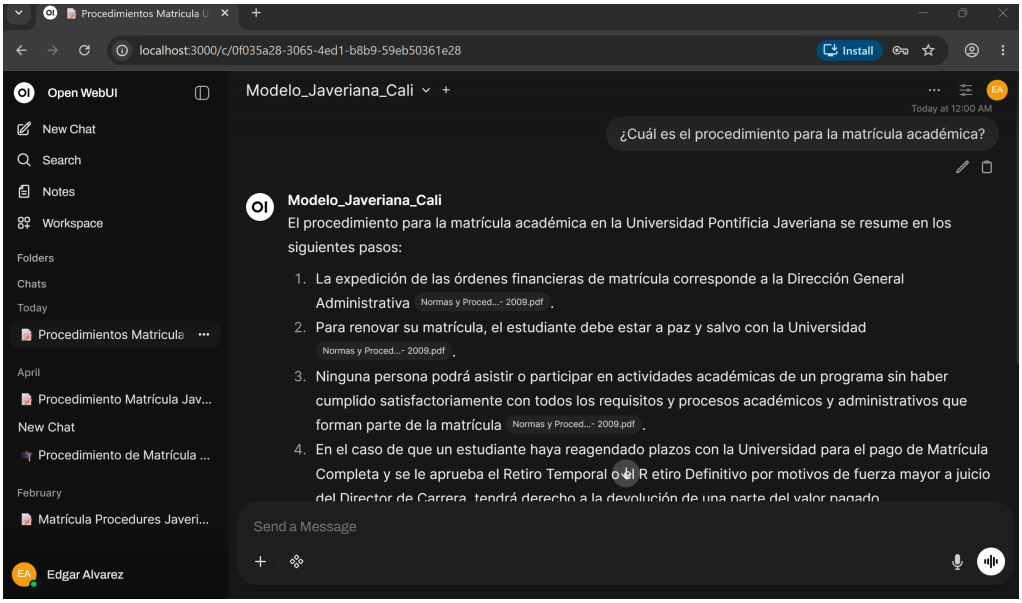

Figura 4.7: Captura de Open WebUI: el usuario formula una pregunta en lenguaje natural y recibe la respuesta del agente RAG con la cita al documento institucional consultado.

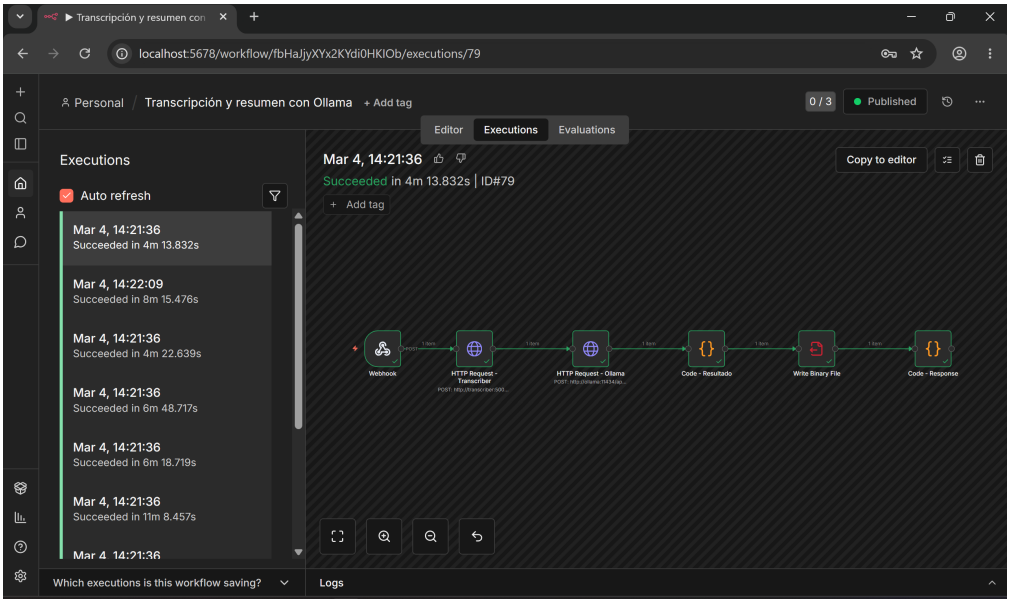

Figura 4.8: Captura del editor de flujos de n8n: el flujo de transcripción muestra los nodos de carga del archivo, llamada al servicio Whisper y postprocesamiento con Ollama. Cada nodo es configurable desde la propia interfaz.

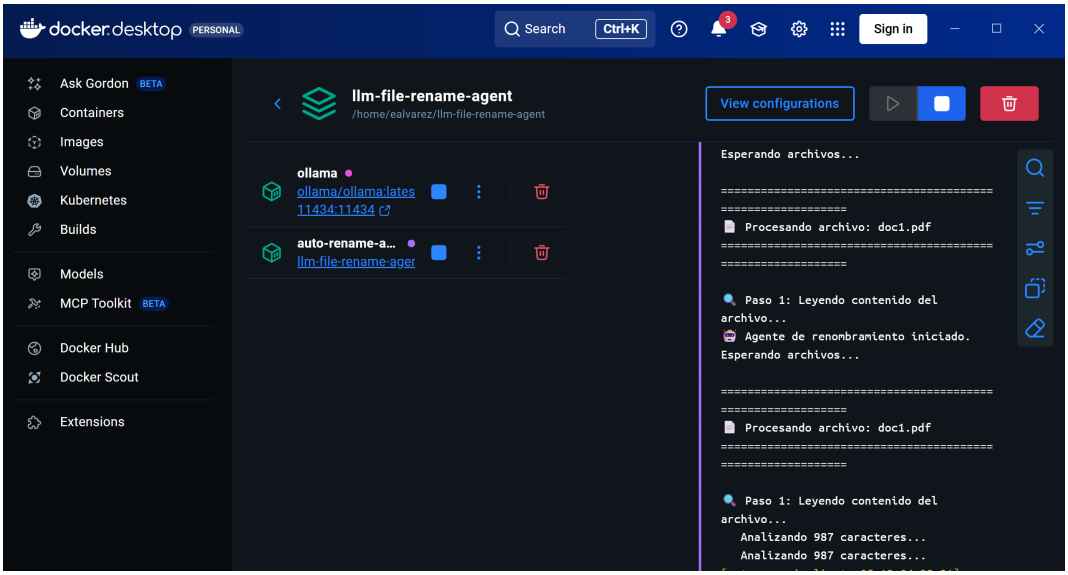

Figura 4.9: Captura del agente de renombramiento: directorios de entrada y salida con archivos antes y después del procesamiento. El nombre asignado por el agente sigue el formato definido en las reglas de nomenclatura.

4.9. Síntesis

Los prototipos desarrollados demuestran la viabilidad técnica de implementar agentes de inteligencia artificial basados en modelos de lenguaje ejecutados localmente para apoyar procesos académico-administrativos. El diseño se orientó a una arquitectura centralizada, administrada por el área de tecnología, en la que el procesamiento se realiza en el backend y las interfaces de usuario el chat de Open WebUI, el editor de flujos de n8n y la pareja de directorios de entrada/salida del agente de renombramiento se eligen para encajar con la forma de trabajo de cada caso de uso.

La implementación en contenedores permite su despliegue en diferentes entornos (servidores locales, máquinas virtuales o infraestructura en la nube institucional), manteniendo el control sobre los datos y facilitando su integración con sistemas existentes. Aunque los tres prototipos son servicios de backend, las capturas de la sección anterior ilustran cómo se ven en operación desde el punto de vista del usuario; los archivos de configuración detallados de cada servicio (`docker-compose.yml`, parámetros de Open WebUI, definición de los nodos de n8n) están disponibles en los repositorios listados en §4.8.2. Estos resultados proporcionan un punto de partida para la evaluación experimental presentada en el Capítulo 5.

Capítulo 5

Evaluación

5.1. Evaluación del desempeño de los prototipos de agentes de IA

Este capítulo presenta la evaluación experimental de los prototipos desarrollados para los casos de uso de Recuperación Aumentada por Generación (RAG), transcripción automática (ASR) y renombramiento semántico.

La evaluación se realizó mediante un enfoque que combina pruebas funcionales, pruebas de rendimiento y la recolección de percepción de utilidad por parte de usuarios administrativos.

El análisis se orienta a las dimensiones definidas en el resultado esperado del proyecto: precisión, eficiencia operativa, facilidad de implementación y adecuación al contexto institucional.

5.1.1. Metodología de evaluación

La evaluación se estructuró en tres etapas:

1. **Pruebas funcionales**: Validación del cumplimiento de cada caso de uso mediante escenarios previamente definidos.
2. **Pruebas no funcionales**: Medición de tiempos de respuesta y comportamiento del sistema bajo carga concurrente.
3. **Evaluación de percepción**: Sesiones guiadas con personal de la Universidad para identificar utilidad percibida y compatibilidad con los procesos institucionales.

Las pruebas de carga y estrés se ejecutaron utilizando Apache JMeter, herramienta ampliamente utilizada para evaluar el desempeño de servicios web bajo diferentes niveles de concurrencia.

Las solicitudes concurrentes se incrementaron progresivamente hasta identificar el punto de saturación del sistema.

5.1.2. Criterios de aceptación

Los criterios de aceptación se definieron antes de realizar las pruebas con el propósito de contar con referentes para interpretar el desempeño observado en cada prototipo. Los valores establecidos no representan estándares universales de calidad, sino rangos considerados razonables para una primera implementación en un contexto institucional.

- **Ejecución correcta de al menos el 90 % de los casos de prueba sin errores críticos.** Este criterio se definió considerando el papel de los agentes como herramientas de apoyo dentro de procesos académico-administrativos, donde todavía existe supervisión humana sobre el resultado. Bajo esta condición, se consideró aceptable que algunos casos requirieran validación o corrección manual, siempre que no comprometieran el funcionamiento general del sistema.
- **Tiempo de respuesta dentro de rangos operativos según el tipo de tarea:**
  - **RAG: menor a 30 segundos por consulta.** En sistemas interactivos, tiempos prolongados de espera pueden afectar la continuidad de la tarea realizada por el usuario. Estudios de usabilidad sitúan alrededor de 10 segundos el punto en que la espera comienza a percibirse como una interrupción relevante (Nielsen, 1993). Dado que en consultas RAG el usuario espera una respuesta basada en recuperación documental e inferencia, se adoptó un margen de tolerancia más amplio para esta primera implementación. El valor también resulta compatible con mediciones reportadas para inferencia local de modelos del rango 7B (Samsi et al., 2023).
  - **ASR: tiempo de procesamiento menor o igual a 2–3 veces la duración del audio.** Este criterio se tomó a partir de resultados publicados para Whisper en hardware de consumo (Radford et al., 2023), donde el tiempo de procesamiento varía según el tamaño del modelo y la capacidad de la GPU utilizada. El rango seleccionado corresponde a configuraciones comparables a las definidas para este trabajo.
  - **Renombramiento: menor a 5 segundos por documento.** Al tratarse de una tarea de procesamiento documental y no de interacción directa con el usuario, el criterio se relaciona principalmente con tiempos razonables de ejecución sobre lotes de archivos. Bajo este rango, el procesamiento de conjuntos amplios de documentos puede realizarse dentro de tiempos compatibles con tareas administrativas periódicas.
- **Estabilidad bajo carga concurrente moderada (hasta 5 usuarios).** Este criterio se definió considerando escenarios piloto con volúmenes limitados de usuarios y configuraciones de infraestructura similares a las descritas para modelos de 7B–14B ejecutados sobre una única GPU.

5.1.3. Entorno de pruebas

Las pruebas se ejecutaron en un entorno controlado sobre un equipo de cómputo personal, cuyas características representan un escenario intermedio de infraestructura institucional. Las especificaciones técnicas del equipo utilizado se presentan en la Tabla 5.1.

Tabla 5.1: Especificaciones del entorno de pruebas

Componente	**Especificación**
Procesador (CPU)	Intel(R) Core(TM) Ultra 9 275HX @ 2.70 GHz
Memoria RAM	32 GB
Almacenamiento	2 TB NVMe SSD
GPU	NVIDIA GeForce RTX 5070 Laptop
Memoria GPU dedicada	8 GB
Memoria GPU compartida	17.9 GB
Memoria GPU total disponible	25.9 GB
Versión del driver GPU	32.0.15.7665

Las pruebas se ejecutaron utilizando modelos de lenguaje locales y herramientas de orquestación descritas en el capítulo anterior, sin el uso de servicios en la nube.

Es importante señalar que los resultados obtenidos en términos de tiempo de respuesta, capacidad de concurrencia y estabilidad del sistema están directamente condicionados por las características del hardware utilizado, particularmente la memoria disponible (RAM y VRAM) y la capacidad de procesamiento de la GPU.

En este sentido, los resultados presentados a continuación deben interpretarse como representativos de un entorno de ejecución local con recursos limitados, alineado con los escenarios de adopción institucional evaluados en este proyecto.

5.1.4. Diseño de las pruebas no funcionales

Las pruebas de rendimiento se realizaron con **Apache JMeter**. En cada prototipo se aumentó gradualmente el número de solicitudes concurrentes hasta identificar el punto en el que el sistema dejaba de responder de forma aceptable. Cada escenario se configuró mediante un *Thread Group* con una rampa de arranque (*ramp-up*) para distribuir la creación de los hilos en el tiempo. Se evaluaron niveles de concurrencia de 5 y 10 solicitudes, y en las pruebas de estrés del agente RAG se amplió el escenario hasta 100 solicitudes concurrentes.

La carga se aplicó sobre el componente principal de cada prototipo. En el caso del agente RAG, JMeter consumió el *endpoint* de chat expuesto por Open WebUI, compatible con la API de OpenAI. Cada solicitud especificaba el modelo, el espacio de trabajo y el conjunto de preguntas definido para la prueba, simulando consultas realizadas por varios usuarios al mismo tiempo. Para el agente de transcripción, las solicitudes se dirigieron al *webhook* del flujo implementado en n8n utilizando un mismo archivo de video como entrada.

El criterio para identificar el punto de saturación se estableció de acuerdo con el comportamiento esperado de cada prototipo. En el agente RAG se consideró alcanzado cuando el tiempo de respuesta superó aproximadamente los seis minutos, ya que a partir de ese punto la interacción dejaba de ser práctica para un usuario. En el agente de transcripción, el límite apareció cuando PostgreSQL dejó de responder y los *workers* de n8n dejaron de procesar nuevas solicitudes. Este comportamiento se

observó al superar aproximadamente diez solicitudes concurrentes, incluso después de aumentar el número de *workers*. Los *scripts* de JMeter utilizados durante las pruebas quedaron versionados en el archivo `README` de los repositorios de cada prototipo.

5.1.5. Resultados por caso de uso

5.1.5.1. Agente RAG

Evaluación funcional

Composición del corpus documental. El repositorio de conocimiento cargado en la base vectorial Qdrant estuvo conformado por **10 documentos institucionales de acceso público**, descargados directamente del sitio web oficial de la Pontificia Universidad Javeriana Cali. Todos los documentos se encuentran abiertos al público en general (manuales, procedimientos, políticas y descripciones de programas académicos), de modo que ningún dato sensible o de circulación interna se utilizó para las pruebas. Los documentos se agruparon en tres categorías temáticas:

- **Política de tratamiento de datos personales**: normativa institucional sobre el manejo, almacenamiento y protección de información personal.
- **Normas y procedimientos de matrícula**: reglamentos académicos que describen los requisitos, etapas y condiciones del proceso de matrícula estudiantil.
- **Información de programas de pregrado**: descripciones de las carreras ofrecidas, incluyendo perfiles de egreso, duración, créditos y requisitos de admisión.

Diseño de las pruebas funcionales. Se formularon **30 preguntas de prueba** distribuidas entre las tres categorías documentales, con el fin de evaluar la capacidad del agente para recuperar información relevante y generar respuestas precisas en lenguaje natural. Las preguntas abarcaron desde consultas directas (e.g., requisitos de matrícula, duración de programas) hasta preguntas que requerían síntesis de información de múltiples fragmentos del mismo documento.

Sobre el tamaño de la muestra. El conjunto de 30 preguntas para el agente RAG y el lote de 50 documentos utilizado en el agente de renombramiento se definieron de acuerdo con el alcance de esta evaluación. Al tratarse de un estudio exploratorio, el propósito no era obtener una muestra estadísticamente representativa, sino comprobar el funcionamiento de los prototipos bajo condiciones controladas. Las preguntas se distribuyeron entre las principales categorías documentales del corpus y cada respuesta fue revisada manualmente utilizando los mismos criterios de evaluación, siguiendo el enfoque propuesto por Lewis et al. (2020). De manera similar, el conjunto de documentos permitió verificar el comportamiento del agente frente a distintos tipos de archivos administrativos sin ampliar innecesariamente el volumen de pruebas.

Las conclusiones obtenidas corresponden al corpus utilizado, al hardware disponible y a las configuraciones implementadas durante la evaluación. Un conjunto de prueba más amplio o la participación de evaluadores independientes permitirían ampliar este análisis en trabajos posteriores.

Metodología de evaluación. La evaluación de las respuestas se realizó mediante **revisión manual** contrastando cada respuesta generada contra el contenido de los documentos fuente. La

revisión manual con criterios fijos sigue la línea propuesta en el trabajo original sobre RAG (Lewis et al., 2020), donde se evalúa cada respuesta contra el documento del que debería provenir. Se emplearon cuatro criterios de evaluación:

1. **Corrección factual**: la información proporcionada en la respuesta coincide con el contenido del documento de referencia.
2. **Fundamentación documental** (*groundedness*): la respuesta se sustenta exclusivamente en el contenido recuperado de la base vectorial, sin incorporar información externa al corpus. La métrica de *groundedness* es estándar en la evaluación de sistemas RAG y se utiliza para detectar alucinaciones del modelo (Lewis et al., 2020).
3. **Atribución de fuente**: la respuesta incluye una cita explícita al documento utilizado, con referencia a la página o sección correspondiente, accesible mediante hipervínculo.
4. **Relevancia**: la respuesta aborda de forma directa la pregunta formulada, sin desviaciones temáticas.

Resultados de la evaluación funcional. La Tabla 5.2 presenta los resultados consolidados de las 30 preguntas de prueba.

Tabla 5.2: Resultados de la evaluación funcional del agente RAG (30 preguntas)

Criterio de evaluación	**Respuestas que cumplen**	**Tasa de cumplimiento**
Corrección factual	30 / 30	100 %
Fundamentación documental	30 / 30	100 %
Atribución de fuente	30 / 30	100 %
Relevancia de la respuesta	30 / 30	100 %

Las 30 respuestas generadas en la ejecución final fueron factualmente correctas, sustentadas en el contenido de los documentos cargados y acompañadas de citas con referencia a la fuente consultada. No se detectaron alucinaciones ni información introducida por el modelo fuera del corpus institucional. Como ejemplo representativo, ante la pregunta *"¿Cuál es el procedimiento para la matrícula académica?"*, el agente respondió citando explícitamente el reglamento correspondiente e indicando los pasos del proceso tal como aparecen en el documento, incluyendo la referencia a la página de origen accesible mediante hipervínculo.

Proceso iterativo previo al 100 % reportado. El resultado de 30/30 respuestas correctas no se obtuvo en una sola ejecución. Las primeras pruebas, antes de los ajustes finales, presentaban dos clases de errores recurrentes: (i) respuestas que mezclaban información de varios documentos cuando la pregunta era ambigua, y (ii) ocasiones en las que el modelo respondía con conocimiento general en lugar de citar el documento, especialmente para preguntas amplias sobre programas académicos. La corrección de cada clase de error implicó ajustes específicos al pipeline:

- Para los errores de mezcla de documentos se ajustó el tamaño y el solape de los *chunks* en la base vectorial (de 512 a 800 tokens con solape de 100), de modo que cada fragmento contuviera contexto suficiente para responder de forma autocontenida.
- Para los errores de respuesta sin cita se reescribió el *prompt de sistema* de Open WebUI exigiendo explícitamente que cada respuesta vaya acompañada de la fuente y que, si la información no está en los fragmentos recuperados, el agente lo declare en lugar de generar información no sustentada.
- Adicionalmente se aumentó de 3 a 5 el número de fragmentos recuperados por consulta (*top-k*), lo que redujo el riesgo de no encontrar el fragmento correcto en preguntas más generales.

El 100 % de la Tabla 5.2 corresponde a la configuración final del pipeline, después de estas iteraciones. Esto explica por qué el resultado supera el criterio de aceptación del 90 % definido en §5.1.2: el criterio se fijó pensando en una primera implementación funcional, y los ajustes iterativos permitieron llegar más lejos sobre un corpus acotado y bien estructurado.

Evaluación de rendimiento bajo carga

Las pruebas de carga se realizaron incrementando progresivamente el número de solicitudes concurrentes hasta identificar el comportamiento del sistema bajo diferentes niveles de demanda.

En condiciones de carga moderada (5 usuarios concurrentes), el sistema mantuvo tiempos de respuesta entre 22 y 36 segundos, con un promedio aproximado de 27 segundos por consulta y una tasa de éxito del 100 %.

En pruebas de estrés posteriores se incrementó el número de solicitudes concurrentes hasta 100 peticiones. Aunque no se registraron errores HTTP, los tiempos de respuesta aumentaron significativamente debido a la formación de colas de inferencia en el modelo de lenguaje.

Los tiempos de respuesta promedio superaron los 400 segundos bajo cargas extremas, lo cual evidencia que el modelo LLM procesa las solicitudes de forma secuencial o con paralelismo limitado.

El análisis de estos resultados permitió identificar que el punto de operación estable del sistema se encuentra aproximadamente entre 4 y 5 solicitudes concurrentes, a partir del cual comienza a observarse degradación progresiva del tiempo de respuesta.

Para escenarios con mayor demanda se propone una arquitectura escalable basada en balanceo de carga y múltiples instancias del modelo de inferencia, como se muestra en la Figura 5.1.

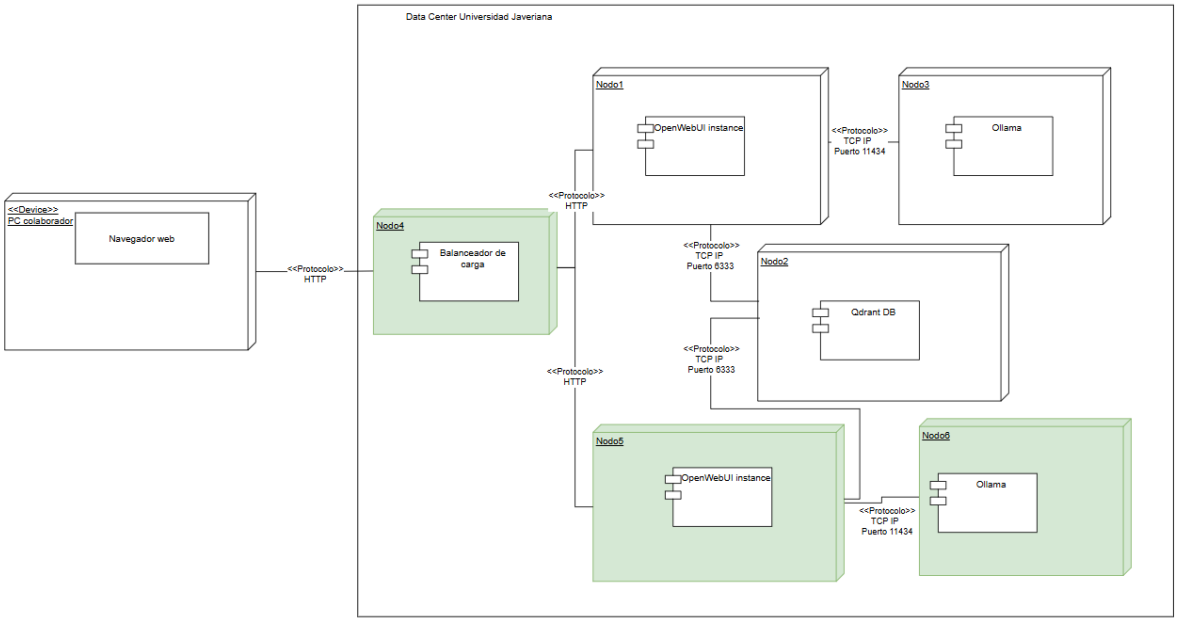

Figura 5.1: Arquitectura escalable para el agente RAG mediante balanceo de carga y múltiples instancias de inferencia

En esta arquitectura el balanceador distribuye las solicitudes entre múltiples nodos de inferencia (Ollama), mientras que la base vectorial Qdrant se mantiene como un servicio compartido.

5.1.5.2. Agente de transcripción automática (ASR)

Evaluación funcional

Composición del conjunto de prueba. Las pruebas funcionales se realizaron sobre un conjunto de **10 archivos multimedia** representativos del formato y duraciones que el agente encontraría en producción. El conjunto incluyó:

- **5 grabaciones cortas (5–15 minutos)** en formato MP4, equivalentes a fragmentos de reuniones de seguimiento con mi directora.
- **5 grabaciones medianas (30–60 minutos)** en formato MP4, equivalentes a reuniones de seguimiento y reuniones con personal admnistrativo de la universidad acerca del proyecto de grado.

Para no exponer información institucional real durante las pruebas, los archivos se obtuvieron de grabaciones de pruebas controladas grabadas específicamente para este trabajo. Los criterios de evaluación funcional fueron tres: (i) que la transcripción se generara sin errores del flujo, (ii) que

el texto producido fuera legible y respetara la puntuación básica, y (iii) que el sistema procesara el formato MP4, sin requerir conversión manual previa. Los 10 archivos se procesaron correctamente bajo estos tres criterios.

Evaluación de rendimiento bajo carga

Las pruebas de carga del agente de transcripción se realizaron utilizando un subconjunto de los archivos previamente evaluados, agrupando varias solicitudes simultáneas con el fin de observar el comportamiento del flujo implementado en n8n bajo diferentes niveles de concurrencia.

Con cinco solicitudes concurrentes, el sistema completó el procesamiento de los archivos presentar errores observables, registrando tiempos de ejecución entre 2.7 y 10.1 minutos por solicitud. Posteriormente, la carga se incrementó hasta diez solicitudes concurrentes. En esta configuración, las peticiones continuaron procesándose correctamente desde el punto de vista funcional, aunque comenzaron a evidenciarse mayores tiempos de espera asociados a la competencia por recursos de procesamiento.

Al superar este nivel de concurrencia se presentaron fallos en componentes utilizados para la orquestación del flujo, particularmente en Redis y PostgreSQL, servicios empleados por n8n para gestionar colas y registrar el estado de las ejecuciones. Bajo la configuración evaluada, el comportamiento observado sugiere que cargas cercanas a diez solicitudes simultáneas representan un punto a partir del cual la estabilidad del sistema empieza a verse afectada.

El aumento en los tiempos de procesamiento resulta consistente con las características del reconocimiento automático de voz, ya que cada solicitud implica el análisis completo del archivo multimedia y una demanda sostenida de recursos computacionales.

A partir de estos resultados, se plantea como alternativa una arquitectura basada en el modo de ejecución en cola (*queue mode*) de n8n, complementada con múltiples *workers* y mecanismos de distribución de carga entre servicios de transcripción. La Figura 5.2 presenta el esquema propuesto.

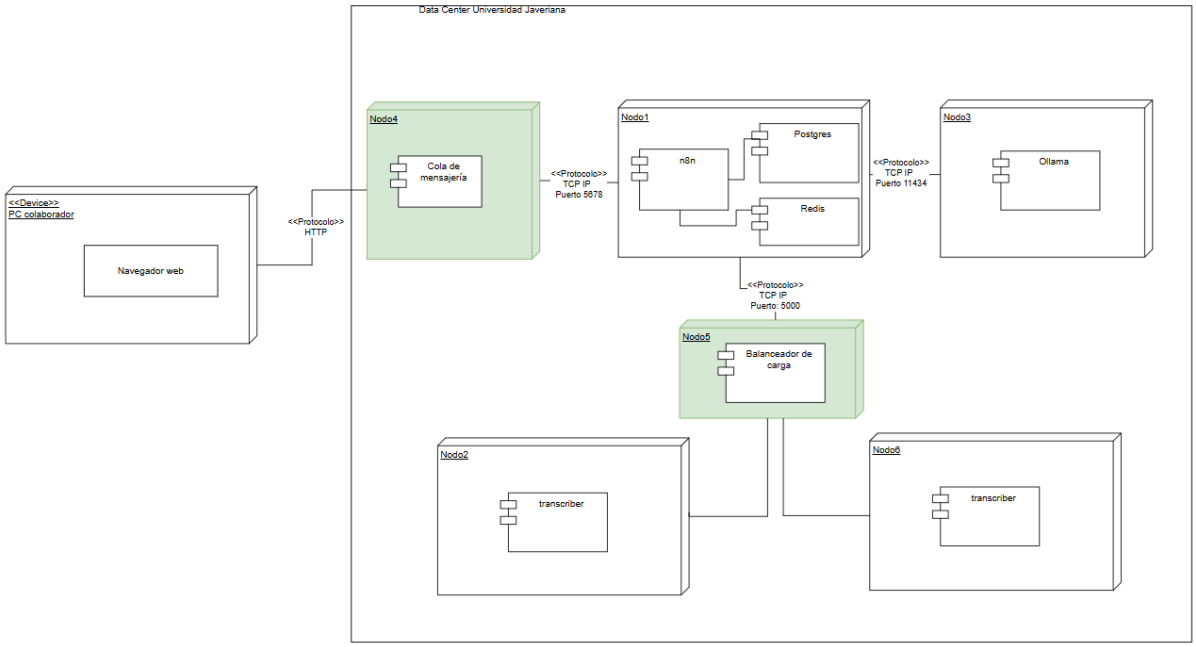

Figura 5.2: Arquitectura escalable del agente de transcripción utilizando cola de trabajos y múltiples workers

En este esquema el flujo de ejecución se organiza de la siguiente manera:

- n8n recibe las solicitudes y las coloca en una cola Redis.
- Los workers de n8n procesan las tareas en paralelo.
- Cada worker invoca el servicio de transcripción basado en Whisper.
- El modelo local genera la transcripción y devuelve el resultado al flujo de trabajo.

Este enfoque permite aumentar la capacidad de procesamiento incrementando el número de workers disponibles.

5.1.5.3. Agente de renombramiento semántico

Evaluación funcional

Composición del conjunto de prueba. Las pruebas funcionales se realizaron sobre un lote de **50 documentos** descargados directamente del sitio web oficial de la Pontificia Universidad Javeriana Cali, todos de **acceso público**. Esto descarta cualquier riesgo de exposición de información sensible institucional durante las pruebas: los documentos están ya disponibles para cualquier

visitante del sitio y se utilizaron únicamente como muestra representativa de los tipos documentales que el agente encontraría en un contexto operativo. El conjunto incluyó las siguientes categorías:

- **Manuales y procedimientos**: instructivos institucionales sobre procesos académicos y administrativos.
- **Políticas**: documentos normativos publicados por la institución (tratamiento de datos personales, reglamentos académicos).
- **Programas académicos**: descripciones públicas de los programas de pregrado, incluyendo perfil de egreso, plan de estudios y requisitos de admisión.

Reglas de renombramiento. El agente fue configurado con un conjunto de reglas de nomenclatura que definen la estructura esperada del nombre generado, incluyendo elementos como tipo de documento, identificación del emisor o destinatario cuando esté disponible en el contenido y referencia temática principal extraída del texto. Estas reglas son **parametrizables**: se cargan desde un archivo de configuración externo y pueden adaptarse a las normas de nomenclatura propias de cada institución (por ejemplo, prefijos por dependencia, formato de fechas, longitud máxima del nombre, codificación específica del archivo institucional). En este trabajo se utilizó una configuración genérica como demostración; el equipo de la Dirección de Tecnologías de Información puede sustituirla por las reglas vigentes del archivo institucional sin modificar el código del agente.

Metodología de evaluación. La validación de los nombres generados se realizó mediante **revisión manual** de cada documento procesado, siguiendo la misma lógica que en el agente RAG: contrastar el resultado del agente contra el contenido fuente con criterios fijos. Para cada archivo se contrastó el nombre asignado por el agente con el contenido del documento, verificando el cumplimiento de dos criterios:

1. **Pertinencia semántica**: el nombre generado refleja con precisión el tipo y contenido del documento.
2. **Conformidad con las reglas**: el nombre sigue la estructura de nomenclatura definida en la configuración del agente.

Resultados de la evaluación funcional. La Tabla 5.3 presenta los resultados del procesamiento de los 50 documentos.

Tabla 5.3: Resultados de la evaluación funcional del agente de renombramiento (50 documentos)

Criterio de evaluación	Documentos que cumplen	Tasa de cumplimiento
Pertinencia semántica	50 / 50	100 %
Conformidad con las reglas de nomenclatura	50 / 50	100 %

Los 50 documentos de la ejecución final recibieron nombres generados correctamente, con pertinencia semántica respecto al contenido y cumplimiento de las reglas de nomenclatura establecidas.

No se registraron casos de renombramiento incorrecto, genérico o inconsistente con el tipo documental.

Proceso iterativo previo al 100 % reportado. Igual que en RAG, este resultado no se obtuvo en una sola ejecución. Las primeras versiones del agente producían dos clases de errores recurrentes: (i) nombres genéricos cuando el documento abordaba múltiples temas (por ejemplo, un reglamento académico que mencionaba varios programas se renombraba con la palabra "reglamento"), y (ii) violaciones puntuales del formato cuando el contenido incluía caracteres especiales o nombres propios largos. La corrección de estos errores requirió:

- Reescribir el *prompt de sistema* del Agente Analizador para forzar la identificación de un único tema principal y descartar referencias secundarias.
- Añadir una etapa de validación posterior al Agente Renombrador que verificara la longitud del nombre y reemplazara caracteres no válidos antes de escribir en disco.
- Iterar sobre 2–3 ciclos de revisión manual del primer lote de prueba hasta que las reglas y los *prompts* produjeran resultados consistentes.

El 100 % de la Tabla 5.3 corresponde, por tanto, a la configuración final ya iterada. Para la tipología de documentos evaluada y bajo las reglas de nomenclatura definidas, el agente produce salidas confiables y reproducibles, pero el resultado depende de un trabajo previo de calibración de *prompts* y reglas que debe rehacerse cuando cambia el tipo de documento o cuando se cambian las reglas institucionales de nomenclatura.

Evaluación de rendimiento

El tiempo promedio de procesamiento fue inferior a **3 segundos por documento**, cumpliendo con el criterio de aceptación definido (menor a 5 segundos). Debido al bajo costo computacional de este proceso, no se observaron variaciones significativas en los tiempos de respuesta bajo cargas moderadas.

5.1.6. Evaluación de percepción con la Dirección de Tecnologías de Información

5.1.6.1. Descripción del stakeholder y participantes

El proceso de evaluación de percepción se realizó en conjunto con la **Dirección de Tecnologías de Información DTI** de la Pontificia Universidad Javeriana Cali, área responsable de la gestión de la infraestructura tecnológica institucional, la administración de servicios y la atención de requerimientos de soporte técnico. Esta unidad fue seleccionada como stakeholder principal por ser el área con mayor capacidad técnica para valorar la viabilidad operativa de los prototipos y por gestionar los procesos internos directamente relacionados con los tres casos de uso evaluados.

Participaron cuatro integrantes del equipo de la DTI:

- **Jefe de Desarrollo e Innovación**: responsable de la dirección estratégica de proyectos tecnológicos dentro del área.

- **Coordinador de Soporte Tecnológico**: encargado de la gestión operativa de los servicios de TI.
- **Ingeniero de infraestructura**: integrante del equipo técnico con experiencia en infraestructura y despliegue de servicios.
- **Líder de exploración de IA local**: integrante de la DTI a cargo de los procesos de adopción de inteligencia artificial en entornos locales dentro de la Universidad. Su perspectiva fue particularmente relevante porque el equipo ya había comenzado a evaluar servidores con GPU para ejecutar LLMs locales, y los resultados de este trabajo se discutieron como insumo directo para esa exploración.

5.1.6.2. Estructura de las sesiones

Se realizaron cuatro sesiones de trabajo con el equipo de la DTI. Cada sesión siguió un formato estructurado que combinó demostración funcional de los prototipos, presentación de resultados experimentales y discusión técnica sobre viabilidad de adopción. La Tabla 5.4 resume el contenido abordado en cada sesión.

Tabla 5.4: Estructura de las sesiones de evaluación con la DTI

Sesión	Prototipo evaluado	Contenido abordado
1	Agente RAG	Demostración funcional del sistema de recuperación de información. Presentación de la arquitectura (Ollama + Qdrant + Open WebUI). Discusión sobre casos de uso aplicables al contexto de la DTI.
2	Agente de transcripción automática (ASR)	Demostración del flujo de transcripción y generación de actas. Presentación de la arquitectura (n8n + Whisper + Redis + PostgreSQL). Discusión sobre integración con procesos administrativos actuales.
3	Agente de renombramiento semántico	Demostración del procesamiento de lotes de documentos. Presentación de resultados funcionales y métricas de rendimiento. Discusión sobre aplicabilidad en el área de archivo institucional.
4	Resultados integrados y estrategia de escalabilidad	Presentación consolidada de resultados de pruebas de carga y estrés para los tres prototipos. Discusión sobre extrapolación de resultados a la infraestructura on-premise existente en la DTI. Revisión de las arquitecturas escalables propuestas para cada caso de uso.

5.1.6.3. Hallazgos por caso de uso

A partir de las sesiones realizadas, el equipo de la DTI identificó aplicaciones concretas para cada prototipo dentro de sus procesos operativos actuales.

Agente RAG — Gestión documental y soporte técnico

El equipo valoró positivamente la capacidad del agente RAG para consultar documentación institucional en lenguaje natural. Se identificó como caso de uso prioritario la mejora en la **gestión de tickets de soporte técnico**: actualmente, parte del tiempo de atención se destina a localizar manuales, procedimientos y políticas internas dispersas en múltiples repositorios. Un agente RAG integrado a la base documental de la DTI podría reducir estos tiempos de consulta, agilizando la resolución de incidencias recurrentes.

Agente de transcripción automática — Generación de actas de reunión

El prototipo de transcripción fue identificado como el de mayor impacto inmediato sobre la carga operativa del personal administrativo. La elaboración manual de actas a partir de grabaciones de reuniones representa una tarea repetitiva y de alto consumo de tiempo. El equipo señaló que la automatización de este proceso mediante el agente ASR permitiría al personal administrativo **redirigir ese tiempo hacia actividades de mayor valor agregado**, manteniendo al mismo tiempo un registro estructurado y consistente de los acuerdos institucionales.

Agente de renombramiento semántico — Clasificación de archivos en el área de archivo

El agente de renombramiento fue relacionado directamente con las actividades del **área de archivo institucional**, que recibe diariamente un volumen significativo de documentos que requieren clasificación y nomenclatura manual antes de su almacenamiento. El equipo consideró que la automatización de este proceso reduciría el tiempo dedicado a tareas repetitivas de organización documental y disminuiría la probabilidad de errores de nomenclatura que dificultan la recuperación posterior de archivos.

5.1.6.4. Consideraciones técnicas planteadas por el equipo

La DTI manifestó interés particular en los resultados de las pruebas de carga y estrés, con el objetivo de **extrapolar el comportamiento de los prototipos a los servidores on-premise** que el área ha adquirido recientemente para explorar la ejecución local de modelos de IA. En este sentido, se plantearon las siguientes consideraciones:

- La necesidad de comprender los límites operativos de cada prototipo en función de la concurrencia esperada en su contexto institucional, a fin de dimensionar adecuadamente los recursos disponibles.
- El interés en las estrategias de escalabilidad propuestas (balanceo de carga para RAG y arquitectura de cola de trabajos para ASR) como guía para planificar una posible implementación en ambiente productivo.
- La relevancia de conocer los requisitos mínimos de infraestructura (VRAM, RAM, almacenamiento) en relación con las especificaciones del servidor institucional disponible, para determinar qué modelos y configuraciones son viables sin inversión adicional inmediata.

Estas consideraciones confirman la pertinencia de los resultados de carga presentados en las secciones anteriores y refuerzan la necesidad de las arquitecturas escalables propuestas como insumo para la toma de decisiones institucional.

5.1.7. Evaluación integrada del desempeño

El análisis experimental se realizó considerando cuatro dimensiones.

Precisión

Los tres prototipos superaron el criterio de aceptación definido (mayor o igual al 90 % de casos sin errores críticos). El agente RAG obtuvo una tasa de cumplimiento del 100 % en los cuatro criterios de evaluación funcional (corrección factual, fundamentación documental, atribución de fuente y relevancia) sobre 30 preguntas de prueba ejecutadas contra un corpus de 10 documentos institucionales. El agente de renombramiento alcanzó igualmente el 100 % de cumplimiento en pertinencia semántica y conformidad con reglas sobre un lote de 50 documentos de tipología variada. El agente ASR generó transcripciones coherentes para todos los archivos multimedia de reuniones de seguimiento procesadas durante las pruebas funcionales.

Eficiencia operativa

Las pruebas de carga evidenciaron comportamientos diferenciados entre los casos de uso:

- RAG: tiempo promedio de 27 segundos por consulta bajo carga moderada.
- ASR: tiempo promedio entre 2.7 y 10 minutos por archivo.
- Renombramiento: menos de 3 segundos por documento.

Las tareas de generación de texto presentan mejor desempeño bajo concurrencia que los procesos intensivos en cómputo, como el reconocimiento automático de voz.

Facilidad de implementación

Los prototipos se desplegaron mediante contenedores Docker y modelos de código abierto, sin requerir licencias comerciales ni infraestructura especializada adicional.

Este enfoque facilita la replicabilidad de los servicios en entornos institucionales y simplifica las tareas de mantenimiento.

Adecuación al contexto institucional

Las sesiones de evaluación con la Dirección de Tecnologías de Información (descritas en la sección anterior) evidenciaron que los tres prototipos tienen aplicabilidad directa en procesos operativos del área. El agente RAG fue vinculado a la reducción de tiempos en la gestión de tickets de soporte; el agente ASR, a eliminar la elaboración manual de actas; y el agente de renombramiento, a la clasificación automatizada de documentos en el área de archivo. En todos los casos, el beneficio central identificado fue la **reducción de tareas manuales repetitivas** y la **centralización del acceso a la información institucional**.

5.1.8. Interpretación de resultados

Los resultados experimentales evidencian que los prototipos desarrollados presentan un funcionamiento correcto y pueden operar de forma estable bajo cargas concurrentes moderadas.

Las tasas de cumplimiento del 100 % corresponden a la versión final de los prototipos, después del proceso de ajustes descrito en las secciones §5.1.4.1 y §5.1.4.3. Durante ese proceso se realizaron cambios en la estrategia de *chunking*, el *prompt* de sistema y el valor de *top-k*, hasta obtener una configuración estable para el corpus utilizado en las pruebas.

Este resultado refleja el comportamiento de los prototipos bajo esas condiciones de evaluación. Un conjunto de documentos más amplio, preguntas diferentes o cambios en la configuración del sistema podrían producir resultados distintos. En consecuencia, las conclusiones corresponden al corpus, al hardware y a las configuraciones empleadas durante este trabajo, y muestran que los prototipos alcanzaron un funcionamiento consistente en el entorno de evaluación definido para la investigación.

No obstante, las pruebas de estrés permitieron identificar límites operativos asociados principalmente a la capacidad de procesamiento de los modelos y a la infraestructura de orquestación.

En el caso del agente RAG, el principal factor limitante corresponde al procesamiento secuencial del modelo de lenguaje, mientras que en el agente de transcripción el límite se encuentra en los componentes de orquestación (cola Redis y base de datos PostgreSQL).

Estos hallazgos permiten establecer criterios de escalabilidad para futuras implementaciones institucionales, tales como el balanceo de carga entre múltiples instancias de inferencia, el uso de arquitecturas basadas en colas de trabajo y la ampliación del número de workers disponibles.

Capítulo 6

Conclusiones

6.1. Conclusiones

El desarrollo del presente proyecto permitió analizar la viabilidad técnica, económica y operativa de implementar agentes de inteligencia artificial basados en modelos de lenguaje grandes (LLMs) en infraestructura local dentro de un entorno universitario. A partir de la ejecución de los cinco objetivos específicos, se establecieron conclusiones que responden directamente a los propósitos de la investigación y que permiten orientar futuros proyectos institucionales relacionados con la adopción de tecnologías de IA.

6.1.1. Conclusiones por objetivo

1. Identificar los requerimientos técnicos necesarios para la ejecución local de LLMs en entornos institucionales. El estudio mostró que la variable más restrictiva en la selección de hardware es la VRAM de la GPU, no la CPU ni la cantidad de RAM del sistema. Para los tres casos de uso de este trabajo, una configuración con GPU de 16–24 GB de VRAM, 32–64 GB de RAM y almacenamiento NVMe es suficiente y permite operar modelos del rango 7B–14B parámetros en formato cuantizado a 4 bits (GGUF). El *runtime* elegido fue Ollama por su sencillez operativa y su compatibilidad de API con OpenAI, lo que facilitó la integración con herramientas como Open WebUI, n8n y AutoGen sin escribir adaptadores. La matriz comparativa de la Tabla 3.2 permitió establecer configuraciones mínimas y recomendadas para cada caso de uso, y los rangos por tamaño de modelo descritos en §3.1.4 dan un mapa claro de cuándo es realista una sola máquina y cuándo se requiere infraestructura empresarial.

2. Analizar los costos de implementación y mantenimiento de agentes de IA con LLMs locales en comparación con soluciones basadas en la nube. La comparación entre ambos enfoques evidenció diferencias en la estructura de costos. La infraestructura local presenta un costo inicial asociado a la adquisición de hardware, pero mantiene valores operativos relativamente estables en el tiempo. Por el contrario, los servicios en la nube no requieren inversión inicial, pero generan costos recurrentes que aumentan proporcionalmente al uso. En escenarios de uso continuo, la proyección a tres años muestra que la infraestructura local representa un costo acumulado menor que los servicios basados en API. Esta diferencia permite considerar la ejecución local como una alternativa sostenible en instituciones con flujos de trabajo constantes.

3. Examinar las implicaciones de privacidad asociadas a la adopción de LLMs en infraestructura propia dentro del contexto universitario. El análisis normativo evidenció que la implementación local favorece el cumplimiento de la Ley 1581 de 2012, el GDPR y las

políticas institucionales de protección de datos, dado que los datos no son enviados a terceros. No obstante, la operación local introduce riesgos propios, como accesos no autorizados, configuraciones inseguras o ausencia de monitoreo. La aplicación de políticas de seguridad, segmentación de red, control de privilegios y auditorías periódicas es necesaria para mitigar estos riesgos. Se concluye que el enfoque local posibilita un mayor nivel de control, siempre que se acompañe de medidas técnicas y administrativas adecuadas.

4. Desarrollar los prototipos de agentes basados en los tres casos de uso definidos (RAG, transcripción automática y renombramiento semántico). Los prototipos desarrollados permitieron validar la capacidad de los modelos locales para ejecutar tareas propias de procesos académico-administrativos. El agente de RAG mostró respuestas coherentes utilizando información contenida en el repositorio documental institucional. El agente de transcripción aplicó modelos ASR locales como Whisper, permitiendo la conversión de audio a texto sin transferir información a la nube. El agente de renombramiento semántico procesó documentos internos aplicando clasificación temática y asignación automática de nombres descriptivos. La implementación permitió verificar que los modelos locales son viables para tareas institucionales que requieren autonomía y conservación de datos.

5. Evaluar el desempeño de los prototipos en términos de eficiencia operativa, percepción de utilidad y alineación con las necesidades institucionales. Los resultados del proceso de pruebas evidenciaron que los prototipos cumplen con los requerimientos funcionales definidos y pueden integrarse en flujos administrativos reales. Las métricas de QA revelaron tasas de acierto aceptables en clasificación semántica, tiempos de respuesta estables en consultas RAG y precisión consistente en transcripción ASR. La evaluación cualitativa, complementada con entrevistas funcionales, indicó que los agentes desarrollados pueden apoyar procesos internos como búsqueda documental, procesamiento de comunicaciones y gestión de archivos académicos. Estos hallazgos permiten proponer lineamientos para futuras implementaciones sostenibles dentro de la institución.

6.1.2. Lecciones aprendidas

Durante el proceso de investigación y experimentación se identificaron las siguientes lecciones:

- La ejecución local de modelos requiere una planeación cuidadosa de recursos técnicos: pequeñas variaciones en memoria o GPU se traducen en diferencias amplias de desempeño. En las pruebas de carga, el agente RAG mantuvo tiempos de respuesta entre 22 y 36 segundos por consulta con cinco usuarios concurrentes, pero los tiempos superaron los 400 segundos al subir a 100 solicitudes concurrentes; el agente de transcripción procesó archivos en 2,7–10 minutos con cinco solicitudes y empezó a fallar en componentes de orquestación al pasar de diez. Estas diferencias no son lineales, y un mismo equipo puede pasar de ser plenamente funcional a inutilizable según el número de usuarios concurrentes que reciba.
- La cuantización y los formatos optimizados permiten reducir costos sin afectar la utilidad práctica de los modelos.

- El análisis de costos debe considerar no solo el hardware, sino también elementos como consumo energético, soporte técnico y actualizaciones.
- Los riesgos de privacidad no desaparecen con la ejecución local, se transforman. La seguridad depende de políticas internas y del mantenimiento continuo.
- La integración de agentes en procesos administrativos requiere trabajo colaborativo entre personal técnico y administrativo para definir adecuadamente los flujos.
- La evaluación con métricas de QA permite identificar limitaciones operativas tempranas, facilitando ajustes en la arquitectura o elección del modelo.
- Los prototipos locales son útiles para procesos con información sensible; sin embargo, los servicios en la nube continúan siendo adecuados para tareas puntuales que requieren modelos de mayor escala.

6.1.3. Conclusión general

La ejecución local de agentes de IA con modelos de lenguaje grandes es una alternativa viable para instituciones que manejan datos sensibles y buscan mantener control sobre su información. Los análisis realizados permiten concluir que esta opción ofrece un equilibrio entre desempeño, costos sostenibles y cumplimiento normativo.

Los resultados de los prototipos y su evaluación indican que los agentes pueden integrarse en procesos académico-administrativos y aportar eficiencia operativa. Finalmente, las recomendaciones establecidas en el proyecto proporcionan una ruta inicial para futuras inicitativas adopción de estos sistemas dentro de la Pontificia Universidad Javeriana Cali.

6.2. Discusión y reflexión final sobre el proceso

El desarrollo de este trabajo implicó una serie de decisiones técnicas y metodológicas que fueron ajustándose a medida que avanzaba el proyecto. La rápida evolución de los modelos de lenguaje, los cambios frecuentes en herramientas de despliegue y las diferencias entre alternativas locales y servicios en la nube hicieron necesario revisar de forma continua algunos supuestos iniciales y replantear decisiones tomadas en etapas tempranas.

Más allá de los resultados obtenidos, el proceso también permitió comprender mejor las limitaciones prácticas asociadas al uso institucional de agentes basados en LLMs, así como las implicaciones de privacidad, infraestructura y mantenimiento que acompañan su adopción. En ese sentido, parte del aprendizaje del proyecto estuvo en traducir conceptos que suelen discutirse de manera abstracta, modelos locales, agentes, RAG o privacidad de datos hacia decisiones concretas de implementación dentro de un contexto institucional.

6.2.1. Si tuviera que decidir hoy: por qué opción me inclinaría

Al finalizar el proyecto, y pensando en un escenario de adopción institucional, el prototipo hacia el que me inclinaría primero sería el **agente RAG**. Esta elección se relaciona con el equilibrio observado entre esfuerzo de implementación, alcance institucional y utilidad práctica. Con una infraestructura relativamente moderada un modelo de 7B ejecutado sobre una sola GPU es posible consultar información documental de distintas dependencias mediante una interfaz conversacional cercana a herramientas que muchos usuarios ya conocen.

El caso de transcripción automática también representa una posibilidad relevante, especialmente en espacios donde reuniones, clases o comités generan grandes volúmenes de audio que luego deben convertirse en registros consultables. Sin embargo, su incorporación institucional requeriría ajustes en los flujos de trabajo y un proceso gradual de apropiación por parte del personal administrativo, de manera que funcione como apoyo a la elaboración de actas y no como sustitución completa de la revisión humana.

El sistema de renombramiento documental presentó retos técnicos diferentes, particularmente por el uso de una arquitectura multiagente basada en AutoGen. Aun así, su aplicación se concentra en un proceso más específico relacionado con la gestión documental institucional. Por esta razón, tendría más sentido considerarlo como una etapa posterior de adopción, una vez existan experiencias previas de uso con otros agentes de mayor alcance transversal.

La discusión entre despliegues locales y servicios en la nube también cambió algunas de las ideas con las que inicié este trabajo. Al comienzo asumía que una migración completa hacia modelos locales representaba el camino más conveniente. Sin embargo, el análisis técnico, económico y operativo mostró un panorama menos binario. Para procesos que involucran datos sensibles como los abordados en este proyecto, la ejecución local ofrece ventajas importantes en términos de privacidad y control de la información. En cambio, para tareas puntuales que demanden modelos considerablemente más grandes o capacidades avanzadas de razonamiento, el uso de servicios externos puede resultar una alternativa más práctica que intentar reproducir localmente infraestructura de gran escala.

Más que una elección definitiva entre un enfoque u otro, el proceso permitió entender que parte de la decisión está en reconocer qué tipo de problema se busca resolver, qué restricciones existen y qué compromisos técnicos está dispuesta a asumir la institución.

6.2.2. Lo más fácil y lo más difícil de configurar

Entre los tres prototipos desarrollados, el caso que requirió menos esfuerzo de configuración fue el sistema RAG implementado con Ollama, Open WebUI y Qdrant. La integración entre estas herramientas permitió disponer de un entorno funcional mediante una configuración compacta en `docker-compose`. En comparación con implementaciones que hace algunos años exigían configurar manualmente servidores de inferencia, bases vectoriales y mecanismos de recuperación, el ecosistema actual facilitó gran parte del trabajo inicial.

La experiencia fue diferente con el sistema de renombramiento documental basado en AutoGen. Aunque el prototipo funcionó, obtener resultados consistentes entre los agentes resultó más complejo

de lo que inicialmente esperaba. Pequeños cambios en los *prompts*, parámetros de temperatura o reglas de interacción entre agentes produjeron diferencias en el comportamiento observado entre ejecuciones. Esto hizo que la etapa de pruebas requiriera múltiples ciclos de ajuste antes de alcanzar una configuración estable para continuar con la evaluación.

El caso intermedio fue el sistema de transcripción construido con n8n y Whisper. La interfaz visual de n8n facilitó la construcción del flujo inicial, pero la integración con servicios de procesamiento y manejo de colas implicó familiarizarse con componentes adicionales como Redis y PostgreSQL. Comprender cómo interactuaban estos elementos dentro del flujo de ejecución tomó más tiempo del previsto y obligó a revisar aspectos de infraestructura que inicialmente no formaban parte del alcance técnico esperado del proyecto.

6.2.3. Lo que aprendí durante el proceso

Al mirar el proyecto en retrospectiva, varios aprendizajes terminaron siendo más importantes de lo que imaginaba al comienzo. Algunos aparecieron durante las pruebas técnicas; otros surgieron al intentar llevar las ideas iniciales a escenarios de uso más cercanos a una institución.

Uno de los cambios más claros tuvo que ver con la forma de entender las limitaciones de un despliegue local. Al inicio asumía que el principal obstáculo estaría en la capacidad de cómputo o en la velocidad de generación de respuestas. Sin embargo, las pruebas mostraron un panorama distinto: factores como la memoria disponible, el consumo de VRAM y el comportamiento bajo concurrencia terminaban condicionando mucho más el funcionamiento del sistema cuando varios usuarios interactuaban al mismo tiempo. Esto cambió la manera en que empecé a pensar la viabilidad técnica de los LLMs locales.

También resultó evidente que construir un prototipo funcional y sostener un sistema institucional son problemas distintos. Durante las pruebas individuales los resultados fueron consistentes, pero las pruebas de carga mostraron comportamientos diferentes cuando aumentaba el número de solicitudes simultáneas. En ese punto, el problema dejó de estar únicamente en la calidad del modelo y empezó a involucrar infraestructura, estabilidad del servicio, monitoreo y capacidad de mantenimiento. Entender esta diferencia ayudó a dimensionar mejor lo que implica pasar de una demostración técnica a un entorno de uso real.

Otro aspecto que cambió mi manera de entender el problema fue el relacionado con la privacidad. Al comenzar el trabajo asumía que ejecutar modelos de lenguaje de forma local resolvía buena parte de los riesgos asociados al manejo de datos. A medida que avanzó la investigación entendí que el despliegue local cambia la ubicación del riesgo, pero no lo elimina. La institución sigue siendo responsable de aspectos como el control de accesos, los registros de actividad, el almacenamiento temporal de información y la integración con otros sistemas. En la práctica, esto significa que adoptar LLMs locales implica tomar decisiones tanto sobre infraestructura como sobre políticas de gobernanza y tratamiento de datos.

Durante el desarrollo del proyecto también quedó claro que la discusión entre ejecutar modelos localmente o utilizar servicios en la nube no tiene una única respuesta. En la Universidad existen antecedentes de ambas aproximaciones y los resultados han sido diferentes según el contexto y las

necesidades de cada iniciativa. Más que inclinar la balanza hacia una de las dos alternativas, este trabajo aporta evidencia para que esa decisión pueda tomarse con un mayor conocimiento de las implicaciones técnicas, económicas y de privacidad.

6.2.4. Trabajo futuro

Los resultados obtenidos durante este trabajo abren varias posibilidades de continuación. Una de ellas consiste en ampliar la capacidad de los prototipos para operar bajo niveles más altos de concurrencia, incorporando mecanismos como balanceo de carga entre múltiples instancias de inferencia y arquitecturas basadas en colas de trabajo. También sería conveniente complementar la evaluación con métricas especializadas, como la tasa de error de palabras (WER) en el caso del agente de transcripción y la participación de evaluadores independientes para valorar las respuestas del agente RAG sobre un corpus de mayor tamaño.

Otra línea de trabajo depende de la evolución de la estrategia institucional alrededor de los LLMs. Si en el futuro se habilita el uso controlado de información confidencial, podrían abordarse procesos de mayor complejidad, como el apoyo a los procesos de acreditación, la clasificación automática de correos, las homologaciones o el desarrollo de un asistente institucional con acceso a memoria organizacional.

Finalmente, el análisis económico podría ampliarse incorporando costos que en este trabajo no fue posible estimar con información institucional, como la reposición de hardware, la dedicación del personal encargado de la operación y una estimación de los beneficios obtenidos con la automatización. Contar con esos datos permitiría construir un modelo de costos más robusto y evaluar con mayor precisión el retorno de la inversión.

Bibliografía

(2022). Iso/iec 27001: Information security management systems. https://www.iso.org/standard/27001.

AI, M. (2024). Mistral 7b documentation. https://mistral.ai.

Anthropic (2023). Is my data used for model training? https://privacy.anthropic.com/en/articles/10023580-is-my-data-used-for-model-training.

Chugh, R., Grandhi, S., Ozturk, P., and Nayak, R. (2022). Robotic process automation: a systematic literature review and research agenda. *Journal of Business Research*, 145:47–54.

Congreso de la República de Colombia (2012). Ley 1581 de 2012: Por la cual se dictan disposiciones generales para la protección de datos personales. Diario Oficial No. 48.587.

Cuzcano, J. S. M., Zúñiga Peña, L. M., Trujillo Robles, P. L., and Sallo Accostupa, V. (2026). Revisión sistemática de la transformación digital en hispanoamérica: retos, tendencias y perspectivas. *Revista InveCom*, 6(1).

Das, B. C., Amini, M. H., and Wu, Y. (2024). Security and privacy challenges of large language models: A survey. *ACM Computing Surveys*, 57(6):130.

DeepSeek-AI (2024). Deepseek-v3 technical report. arXiv preprint. arXiv:2412.19437.

DeepSeek-AI (2025). Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv preprint. arXiv:2501.12948.

Dennstädt, F., Hastings, J., Putora, P. M., Schmerder, M., and Cihoric, N. (2025). Implementing large language models in healthcare while balancing control, collaboration, costs and security. *npj Digital Medicine*, 8:143.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. (2022). LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems (NeurIPS 2022)*.

Dorca Josa, A. and Bleda-Bejar, M. (2024). Local llms: Safeguarding data privacy in the age of generative ai. a case study at the university of andorra. In *ICERI2024 Proceedings*, pages 7879–7888. IATED.

European Union (2016). General data protection regulation (gdpr): Regulation (eu) 2016/679. Official Journal of the European Union, L119.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2023). GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *Proceedings of the 11th International Conference on Learning Representations (ICLR 2023)*.

Gal, U. (2023). Chatgpt is a data privacy nightmare. if you've ever posted online, you ought to be concerned. https://www.sydney.edu.au/news-opinion/news/2023/02/08/chatgpt-is-a-data-privacy-nightmare.html.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint. arXiv:2312.10997.

Gerganov, G. and llama.cpp contributors (2025). llama.cpp: Llm inference in c/c++. https://github.com/ggerganov/llama.cpp. Consultado en febrero de 2026.

González, J. and Martínez, D. (2023). Implementación de asistentes conversacionales en servicios estudiantiles: Caso uab. *Revista Iberoamericana de Educación Superior*, 14(1):101–115.

Google (2025). Gemini apps privacy notice. https://support.google.com/gemini/answer/13594961.

Grattafiori, A., Dubey, A., et al. (2024). The llama 3 herd of models. arXiv preprint. arXiv:2407.21783.

Grünig, M. et al. (2026). Implementation and user evaluation of an on-premise large language model in a german university hospital setting: Cross-sectional survey. *JMIR AI*, 1:e84362.

Hernández, J. E. (2026). Estos son los cálculos sobre tasas de interés, pib, dólar y petróleo para colombia en 2026, según fedesarrollo. *Portafolio*. Consultado el 10 de julio de 2026.

Hugging Face (2025). Transformers: State-of-the-art machine learning for pytorch, tensorflow, and jax. https://huggingface.co/docs/transformers. Consultado en febrero de 2026.

ICT&Health (2025). What happens to the data that doctors enter in chatgpt? https://icthealth.org/news/what-happens-to-the-data-that-doctors-enter-in-chatgpt.

International Organization for Standardization (2017). ISO/IEC 29134:2017 — information technology — security techniques — guidelines for privacy impact assessment. https://www.iso.org/standard/62289.html.

Investopedia (2024a). Capital expenditure (capex) definition, formula, and examples. https://www.investopedia.com/terms/c/capitalexpenditure.asp. Consultado en febrero de 2026.

Investopedia (2024b). Operating expense (opex) definition and examples. https://www.investopedia.com/terms/o/operating_expense.asp. Consultado en febrero de 2026.

Kay, S. (2020). Uso de herramientas y modelos de procesos empresariales. https://www.processmaker.com/es/blog/business-process-modeling/#:~:text=Un%20modelo%20de%20proceso%20empresarial,cuentas%20en%20toda%20la%20organizaci%C3%B3n.

Lanham, M. (2025). *AI Agents in Action*. Manning Publications.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*.

Li, J. (2023). Security implications of ai chatbots in health care. *Journal of Medical Internet Research*, 25:e47551.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. (2024). AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of the 7th Conference on Machine Learning and Systems (MLSys 2024)*.

Lindholm, J. and Kallio, A. (2023). Automation of internship administration processes using rpa and ai agents at the university of helsinki. *Journal of Higher Education Innovation and Technology*, 5(1):34–47.

Meta AI (2024). Llama 3 model card and technical specifications. https://ai.meta.com/llama.

National Institute of Standards and Technology (2023). Artificial intelligence risk management framework (ai rmf 1.0). https://www.nist.gov/itl/ai-risk-management-framework. Accessed: 2025-01.

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann, San Francisco, CA.

NVIDIA (2023). Nvidia gpu architecture performance guide. https://www.nvidia.com.

Ollama (2025). Ollama: Get up and running with large language models locally. https://ollama.com. Consultado en febrero de 2026.

Open WebUI (2025). Open webui: User-friendly webui for llms (formerly ollama webui). https://openwebui.com. Consultado en febrero de 2026.

OpenAI (2026). Privacy policy. https://openai.com/policies/privacy-policy.

Organisation for Economic Co-operation and Development (2019). Oecd principles on artificial intelligence. https://oecd.ai/en/ai-principles. Accessed: 2025-01.

OWASP Foundation (2025). OWASP top 10 for large language model applications. https://owasp.org/www-project-top-10-for-large-language-model-applications/. Consultado en febrero de 2026.

Pontificia Universidad Javeriana Cali (2021). Modelo de procesos de la pontificia universidad javeriana cali. https://javerianacaliedu.sharepoint.com/sites/Sgc/SitePages/Inicio.aspx.

Qdrant (2025). Qdrant: Vector database for ai applications. https://qdrant.tech. Consultado en febrero de 2026.

Radas, J., Risse, B., and Vogl, R. (2024). Building unigpt: A customizable on-premise llm-solution for universities. In *EPiC Series in Computing, Proceedings of EUNIS 2024*, volume 105, pages 108–116. EasyChair.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*.

Russell, S. and Norvig, P. (2021). *Artificial Intelligence: A Modern Approach*. Pearson, Upper Saddle River, NJ, 4 edition.

Samsi, S., Zhao, D., McDonald, J., Li, B., Michaleas, A., Jones, M., and Gadepally, V. (2023). From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy (S&P)*, pages 3–18.

Tan, Y. S., Lim, C. J., and Wong, L. M. (2023). Intelligent email routing system for student services at national university of singapore. *International Journal of Educational Technology Integration*, 11(2):88–102.

UNESCO (2021). Recommendation on the ethics of artificial intelligence. https://unesdoc.unesco.org/ark:/48223/pf0000381137.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, pages 6000–6010. Curran Associates, Inc.

vLLM Project (2025). vllm: A high-throughput and memory-efficient inference and serving engine for llms. https://github.com/vllm-project/vllm. Consultado en febrero de 2026.

Voigt, P. and Von dem Bussche, A. (2017). *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer International Publishing.

Zhang, G., Jin, Q., Zhou, Y., Wang, S., Idnay, B., Luo, Y., and Peng, Y. (2024). Closing the gap between open source and commercial large language models for medical evidence summarization. *npj Digital Medicine*, 7:239.