Revista Innovación y Software
Vol. 7, No. 2, Mes Septiembre - Febrero, 2026
ISSN: 2708-0935
Pág. 75-87
https://revistas.ulasalle.edu.pe/innosoft

Esta obra está bajo una Licencia Creative Commons Atribución 4.0 Internacional.

Tipo de artículo: Artículos originales
Temática: Inteligencia artificial
Recibido: 26/08/2026 | Aceptado: 18/09/2026 | Publicado: 30/09/2026

Identificadores persistentes:
DOI: 10.48168/innosoft.s30.a408
ARK: ark:/42411/s30.a408

Diseño y verificación funcional de un prototipo RAG híbrido para consulta documental técnica

Design and functional verification of a hybrid RAG prototype for technical document consultation

Moisés Alexander Félix Romero[1*]

[1]Universidad Católica Sedes Sapientiae, Perú. 2021102378@ucss.pe

[*]Autor para correspondencia: 2021102378@ucss.pe

Resumen
La consulta de documentación técnica mediante modelos de lenguaje requiere distinguir la recuperación de fuentes, la generación de respuestas y la verificación de su contenido. Este trabajo presentó el diseño y la verificación funcional de un prototipo web de generación aumentada por recuperación, orientado a consultas documentales en español. Se adoptó un estudio de desarrollo tecnológico con inspección de código, revisión de evidencias de una demostración académica y ejecución de pruebas aisladas. La implementación integró FastAPI, PostgreSQL con pgvector y servicios de Gemini; combinó recuperación densa y búsqueda léxica mediante fusión por rangos recíprocos, conservó localizadores documentales y contempló reordenamiento opcional. La suite existente produjo 18 pruebas aprobadas y ninguna fallida. Ocho escenarios complementarios de caracterización permitieron examinar la abstención, la degradación a búsqueda léxica y el tratamiento de respuestas y registros. Se identificaron diferencias entre disponer de fuentes recuperadas y verificar las afirmaciones generadas, así como limitaciones en la propagación de la abstención textual y en la confirmación de persistencia. Las evidencias documentales previas ilustran una ingesta, una consulta y un fallo controlado del proveedor sobre material ficticio. Estos resultados corresponden a verificación de software y no estiman precisión semántica, latencia productiva ni impacto en tiempos de atención. La contribución consiste en una descripción trazable del artefacto y de sus límites, acompañada de un protocolo para su posterior evaluación documental y operativa.

Palabras claves: asistencia técnica; generación aumentada por recuperación; recuperación de información; trazabilidad documental; verificación de software.

Abstract
Technical document consultation through language models requires distinguishing source retrieval, answer generation, and content verification. This paper presented the design and functional verification of a web-based retrieval-augmented generation prototype for Spanish-language document queries. The study followed a technological development approach combining source-code inspection, review of evidence from an academic demonstration, and execution of isolated software tests. The implementation integrated FastAPI, PostgreSQL with pgvector, and Gemini services. It combined dense retrieval and lexical search through reciprocal rank fusion, preserved document locators, and supported optional reranking. The existing test suite completed with 18 passing tests and no failures. Eight additional characterization scenarios examined abstention, fallback to lexical retrieval, and the handling of generated responses and execution logs. The analysis identified a distinction between returning retrieved sources and verifying generated claims, together with limitations in the propagation of textual abstention and confirmation of log persistence. Previously recorded demonstration evidence illustrates document ingestion, a query response, and a controlled provider failure using fictional material. These findings

concern software verification and do not estimate semantic accuracy, production latency, or reductions in service resolution time. The contribution is a traceable description of the implemented artifact and its limitations, supported by an explicit protocol for subsequent document-level and operational evaluation. The study provides a bounded basis for further development without treating automated test success as evidence of real-world effectiveness.

Keywords: *document traceability; information retrieval; retrieval augmented generation; software verification; technical assistance.*

Introducción

La consulta de manuales y procedimientos exige localizar información pertinente y comprobar su procedencia antes de aplicarla. Un modelo de lenguaje puede producir una respuesta fluida sin que esa respuesta corresponda a una versión documental determinada. La generación aumentada por recuperación, conocida como RAG por sus siglas en inglés, incorpora una memoria documental externa al proceso generativo [1]. Esta separación permite actualizar el corpus y mostrar evidencia recuperada, pero no garantiza por sí misma la corrección de cada afirmación.

La literatura distingue configuraciones RAG básicas, avanzadas y modulares, con diferentes mecanismos de preparación, recuperación y organización de la generación [2]. Estas categorías describen decisiones arquitectónicas; no constituyen una escala de calidad demostrada para cualquier aplicación. En un prototipo, una recuperación más compleja puede añadir dependencias y puntos de fallo, por lo que resulta necesario describir tanto el recorrido esperado como las condiciones bajo las cuales el sistema deja de cumplirlo.

Entre los antecedentes peruanos, Manchay García desarrolló un prototipo local para consultas de mesa de ayuda mediante LlamaIndex, ChromaDB y Gemma3, acompañado de evaluación con Ragas [3]. Su configuración ofrece un contraste con soluciones que delegan embeddings y generación a una API externa. En un contexto académico latinoamericano, Guerra, Monti y Fernández presentaron un sistema para consultar programas de asignaturas en PDF mediante recuperación semántica y enrutamiento por carrera [4]. Estos trabajos apoyan la pertinencia de estudiar asistentes documentales, aunque sus resultados no pueden trasladarse a otro corpus, proveedor o proceso de atención.

La fusión por rangos recíprocos, o RRF, permite combinar listas ordenadas sin sumar directamente puntuaciones de distinta escala [5]. El reordenamiento con modelos que procesan conjuntamente consulta y pasaje representa otra alternativa para refinar los candidatos [6]. Sin embargo, decidir si estos componentes mejoran una aplicación requiere evaluar recuperación y generación por separado; RAGAs ilustra esa necesidad de evaluación diferenciada [7]. Una prueba de API satisfactoria no reemplaza una medición de respuesta sustentada.

El problema abordado en este artículo es la correspondencia entre el diseño de un asistente documental, su implementación y la evidencia que permite afirmar que sus controles funcionan. Se presenta un prototipo académico inspirado en la consulta técnica de una mesa de ayuda, sin incorporar documentación operativa ni participantes de empresas de telecomunicaciones. El objetivo es describir el artefacto implementado y verificar sus comportamientos observables mediante inspección y pruebas aisladas. Se plantean dos preguntas: ¿qué mecanismos de recuperación, trazabilidad y manejo de fallos implementa el prototipo?, y ¿qué comportamientos y límites pueden comprobarse sin una evaluación operativa?

La contribución es una caracterización técnica reproducible de la integración, de sus estados de respuesta y de los límites de la evidencia disponible. No se propone un nuevo algoritmo de recuperación ni se declara superioridad frente a otras arquitecturas. El alcance comprende una fase de desarrollo y verificación; la precisión sobre un corpus representativo y el impacto en tiempos de atención corresponden a una fase posterior.

Materiales y metodología computacional

Diseño y fuentes de evidencia

El estudio se orienta al diseño y examen de un artefacto de software, en consonancia con la distinción entre construcción de soluciones y evaluación de su utilidad planteada en la investigación de diseño [8]. Se aplicaron tres procedimientos: inspección de los módulos y del esquema de datos, revisión de las capturas de una demostración académica previa y ejecución local de pruebas automatizadas. No se realizó un experimento con analistas ni una comparación de productividad.

La unidad de verificación es un comportamiento del programa bajo una entrada y un estado controlados. La unidad documental de la demostración previa es un PDF ficticio de una página, según su registro de ingesta. La revisión del código utilizó una copia local preservada del paquete del prototipo; las pruebas descritas no modificaron la implementación. Para identificar la versión examinada se calcularon huellas SHA-256 de sus archivos, incluidas en el material de verificación.

Se distinguen cuatro niveles de evidencia: configuración declarada en código; observación visual previa; resultado de una aserción automatizada; y eficacia operativa. Solo los tres primeros se encuentran disponibles. Una variable de entorno con un modelo predeterminado no acredita que ese modelo se haya utilizado en una captura histórica; del mismo modo, un resultado simulado no acredita disponibilidad de la API externa.

Revista Innovación y Software
Vol. 7, No. 2, Mes Septiembre - Febrero, 2026
ISSN: 2708-0935
Pág. 75-87
https://revistas.ulasalle.edu.pe/innosoft

Arquitectura y persistencia

El prototipo integra una interfaz HTML y JavaScript servida por FastAPI, un servicio de recuperación y generación en Python y una base PostgreSQL con pgvector. La ingesta utiliza pypdf para extraer texto por página [9]. Los PDF sin texto extraíble se rechazan; el reconocimiento óptico de caracteres no está implementado. La figura 1 representa la organización de los componentes y sus dependencias.

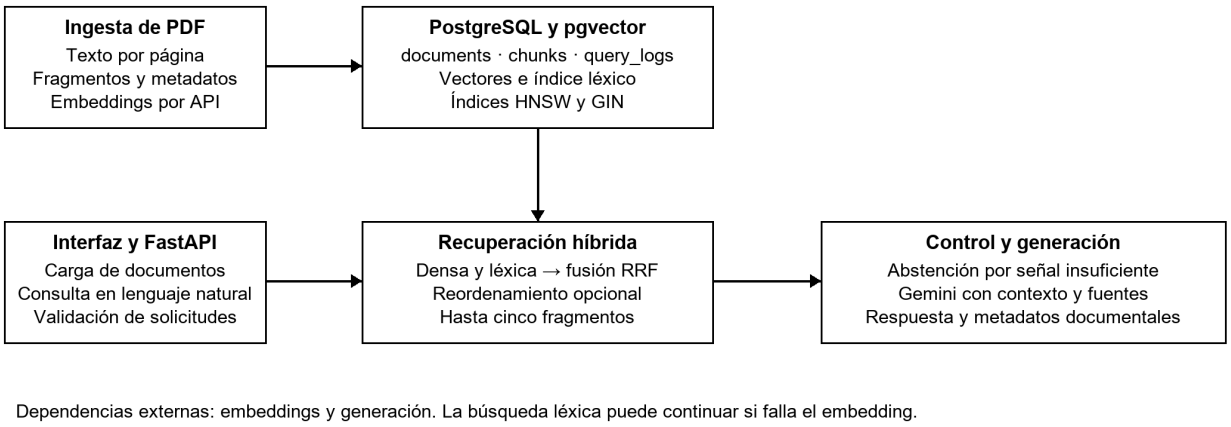


Figura 1. Arquitectura del prototipo inspeccionado. Elaboración propia a partir de los módulos de ingesta, recuperación y respuesta.

Cada documento se almacena en documents; sus fragmentos se conservan en chunks con nombre relacionado, versión, página y un localizador aproximado de párrafo. El servicio solicita representaciones vectoriales a Gemini, cuyo mecanismo de embeddings permite configurar tareas de recuperación [10]. La dimensión predeterminada de 1536 componentes coincide con la columna vectorial del esquema. Cambiarla requiere ajustar de manera consistente la configuración y la base; la verificación local no volvió a generar embeddings en línea.

La extensión pgvector habilita consultas de similitud vectorial en PostgreSQL [11]. El esquema crea un índice HNSW, basado en grafos navegables jerárquicos [12], y un índice GIN para búsqueda de texto completo. La existencia del índice en el esquema no demuestra que el planificador lo utilice en cada consulta: ese comportamiento requiere examinar planes de ejecución y datos reales. La tabla query_logs conserva pregunta, respuesta, abstención y latencia, pero no el conjunto completo de evidencias ni un identificador de usuario.

Revista Innovación y Software
Vol. 7, No. 2, Mes Septiembre - Febrero, 2026
ISSN: 2708-0935
Pág. 75-87
https://revistas.ulasalle.edu.pe/innosoft

Tabla 1. Configuración identificada en el código

Elemento	Valor o comportamiento	Alcance
Fragmentación	1400 caracteres; solapamiento de 180	Por página, con localizador aproximado
Embeddings	gemini-embedding-001; 1536 dimensiones	Valores predeterminados, no medición en línea
Generación	gemini-2.5-flash; temperatura 0	Modelo configurable mediante entorno
Recuperación	Hasta 20 candidatos por canal	Unión, RRF y truncamiento a 20
Selección final	Hasta 5 fragmentos	Reordenamiento opcional sobre candidatos retenidos
Abstención	MIN_RETRIEVAL_SCORE = 0,35	Umbral provisional de similitud densa
Proveedor	3 intentos; esperas de 1 y 2 segundos	Fallos transitorios específicos y timeouts

Fragmentación y recuperación

La extracción conserva el número de página del PDF. Dentro de cada página, el texto se divide utilizando separaciones de párrafo y límites de caracteres; cuando un segmento supera el máximo, se generan ventanas con solapamiento. Este procedimiento no equivale a fragmentación por tokens ni a una jerarquía padre e hijo. El índice de párrafo procede del texto extraído y puede incluir contenido solapado del párrafo anterior, por lo que no debe interpretarse como un localizador editorial exacto.

Para cada pregunta se intenta calcular un embedding de consulta y recuperar vecinos por distancia de coseno. El canal léxico emplea plainto_tsquery y ts_rank_cd con la configuración simple de PostgreSQL [13]. No implementa BM25. En el código revisado, las consultas densa y léxica se ejecutan secuencialmente; no se atribuye ejecución paralela. Si la obtención del embedding falla, el recuperador continúa con el canal léxico y registra una advertencia en el registro de la aplicación.

Sea c un fragmento y $r_j(c)$ su posición, comenzando en uno, en la lista del canal j. La ecuación 1 representa la fusión implementada, que suma únicamente los canales donde aparece c [5]. La unión contiene como máximo 40 candidatos cuando ambos canales devuelven 20. Tras ordenarlos por RRF, se retienen como máximo 20 para el reordenamiento opcional y se devuelven cinco. Esta precisión distingue el tamaño de la unión del número

efectivo de pares que recibe el reordenador.

$$RRF(c) = \sum_{j} \frac{1}{60 + r_j(c)} \tag{1}$$

Con L candidatos por canal y U fragmentos distintos, U $\leq$ 2L, la acumulación de RRF mediante diccionario requiere tiempo esperado O(L) y espacio O(U); ordenar la unión requiere O(U log U). Estas cotas describen la fusión, no el costo de la consulta a la base, del cálculo de embeddings ni de la generación. No se infiere una latencia determinada a partir del uso de un índice aproximado.

Respuesta y manejo de dependencias

El generador recibe la pregunta y un contexto formado por los fragmentos seleccionados, junto con documento, versión, página y párrafo cuando están disponibles. La integración utiliza el SDK de Google Gen AI [14]. La instrucción del sistema solicita responder en español, citar las afirmaciones técnicas y abstenerse ante evidencia insuficiente. Se trata de una instrucción al modelo; la implementación no añade una comprobación semántica posterior de todas las afirmaciones.

La política previa a la generación se basa en tres condiciones. Si no hay fragmentos, se produce abstención. Si existe algún puntaje denso positivo y el máximo es inferior a 0,35, también se produce abstención. Si no hay puntajes densos positivos, el flujo puede continuar cuando existe puntuación léxica positiva. El umbral no constituye una probabilidad de corrección ni ha sido calibrado con consultas etiquetadas. La rama léxica mantiene una posibilidad de respuesta, pero no garantiza relevancia semántica.

Las llamadas al proveedor admiten hasta tres intentos en total ante timeouts y códigos transitorios seleccionados. El tiempo de espera del cliente se configura en 20 000 milisegundos por solicitud; no representa un límite global de toda la operación, porque pueden acumularse intentos y etapas. La API transforma la indisponibilidad conocida en una respuesta HTTP 503 y puede indicar Retry-After. El acceso por token es configurable y no equivale a autenticación corporativa con roles.

Procedimiento de verificación

La suite existente se ejecutó el 23 de octubre de 2026 en Windows con Python 3.12.14, pytest 8.3.4, FastAPI 0.115.6, Pydantic 2.10.4 y HTTPX 0.28.1. Se preservaron sus 18 casos sin alterar el código de producción. Las pruebas de API utilizan TestClient, mecanismo documentado por FastAPI [15], y sustituciones controladas mediante monkeypatch, disponibles en pytest [16]. Se exportó un informe JUnit con el resultado de cada caso.

La ejecución no desplegó Docker ni abrió una conexión a una base o modelo real.

Se añadieron ocho escenarios de caracterización en un archivo separado. Para cada uno se fijaron los fragmentos recuperados o el fallo de persistencia, se sustituyó el cliente generativo por una respuesta predeterminada y se observó el estado devuelto. Las salidas del generador utilizadas en estos escenarios son datos sintéticos de prueba y no respuestas obtenidas de Gemini. Se registraron los resultados en JSON, sin utilizar sus tiempos como medida de rendimiento.

La revisión visual de las capturas se realizó de forma independiente. Se transcribieron únicamente los campos visibles y se evitó asumir que las tres capturas pertenecían a una misma ejecución o configuración. La captura de consulta contiene versión nula aunque la pantalla de carga muestra una versión introducida; por ello no se utilizó ese material para acreditar continuidad de versiones entre ambos recorridos.

Resultados y discusión

Resultados de la suite existente

Las 18 pruebas finalizaron aprobadas, sin fallos. El ejecutor informó 2,13 segundos y una advertencia de obsolescencia de una interfaz de AnyIO utilizada por Starlette. Ese tiempo corresponde a la suite aislada, no al tiempo de respuesta del asistente. La tabla 2 organiza los casos por archivo y explicita qué parte de la solución verifican.

Tabla 2. Pruebas automatizadas existentes ejecutadas

Grupo	**Casos aprobados**	**Comportamientos cubiertos**
API	7 de 7	Salud, interfaz, validación de pregunta, token de carga, errores y versión enviada por formulario
Lógica RAG	5 de 5	Tamaño de fragmento, metadatos, conversión de puntajes, recuperación léxica alternativa y ocultación de credenciales
Reintentos	4 de 4	Recuperación tras fallo temporal, agotamiento, error permanente y timeout
Matriz de evaluación	2 de 2	Rechazo de mediciones incompletas y de un número insuficiente de consultas

El resultado permite afirmar que las aserciones incluidas se cumplen en ese entorno. No permite estimar cobertura de código, tasa de respuestas correctas ni proporción de incidencias resueltas. En particular, los casos de la matriz comprueban el rechazo de entradas incompletas o insuficientes; no demuestran que un candidato arquitectónico haya obtenido mejores resultados medidos.

Caracterización de estados y trazabilidad

Los escenarios adicionales confirmaron la abstención previa a la generación cuando no existen fragmentos y cuando la señal densa positiva resulta insuficiente. También reprodujeron la posibilidad de continuar con coincidencias exclusivamente léxicas. La tabla 3 recoge las ocho observaciones; cada fila describe un caso controlado, no una frecuencia esperada de ocurrencia.

Tabla 3. Observaciones de los escenarios complementarios

Caso	Condición controlada	Observación
A01	Lista de fragmentos vacía	Abstención; no se invoca el generador
A02	Similitud densa 0,20 y puntuación léxica positiva	Abstención previa a generar
A03	Sin señal densa positiva y con coincidencia léxica	Se genera y devuelve estado answered
A04	El generador devuelve texto vacío	Se inserta un mensaje de imposibilidad, pero permanece answered y abstained=false
A05	El generador expresa abstención en texto	Permanece answered y abstained=false
A06	Dos fuentes recuperadas y una mencionada en la respuesta	citations devuelve las dos fuentes; no verifica correspondencia por afirmación
A07	Fallo simulado de conexión al registrar	La función no propaga el fallo ni confirma que el registro se guardó
A08	Sin señal densa positiva ni coincidencia léxica	Abstención; no se invoca el generador

La discrepancia entre texto y estado en A04 y A05 es relevante para cualquier consumidor de la API. Si un tablero cuenta answered como respuesta útil, podría contabilizar como éxito una salida que reconoce no poder contestar. Una evolución del diseño debería distinguir respuesta generada, respuesta vacía y abstención, con

criterios explícitos y pruebas correspondientes. Esta recomendación se deriva del comportamiento observado y no se presenta como una corrección ya implementada.

A06 muestra una segunda distinción: devolver los documentos utilizados para construir el contexto no demuestra que cada afirmación esté apoyada por ellos. La lista citations es trazabilidad de recuperación. Para verificar sustento se necesita comparar afirmaciones y pasajes, de forma manual o mediante un método evaluado. Del mismo modo, el localizador aproximado de párrafo requiere cautela cuando la extracción altera separaciones o el fragmento contiene solapamiento.

A07 indica que la disponibilidad de la respuesta se prioriza frente a la confirmación del registro. Esa decisión evita que un fallo de bitácora interrumpa necesariamente la interacción, pero deja un límite de auditoría: la ausencia de excepción no demuestra persistencia. No corresponde describir el registro actual como inmutable, completo o suficiente para reconstruir todas las decisiones de recuperación.

Alcance de la demostración documental

Las capturas previas aportan tres observaciones puntuales. La pantalla de ingesta informa document_id=4, pages_with_text=1 y chunks_created=2 para un documento de prueba. La pantalla de consulta muestra una respuesta sobre ROUTER-TEST-01 con la dirección ficticia 192.168.1.1 y una referencia a la página 1. La tercera pantalla presenta un mensaje controlado de indisponibilidad o agotamiento de cuota. Estos registros son evidencia visual conservada del desarrollo; no se reprodujeron como un nuevo ensayo en línea durante esta verificación.

El ejemplo permite ilustrar el flujo de interacción, pero no permite calcular F1, Context Recall, fidelidad ni latencia P95. Tampoco verifica la vigencia de documentación operativa, la calidad de múltiples versiones o el desempeño bajo concurrencia. El material ficticio delimita el caso y evita atribuir resultados a una organización o a personal que no participó en la evaluación.

Relación con los antecedentes y límites

Frente al prototipo local de Manchay García [3], esta implementación conserva la base documental en PostgreSQL pero depende de un proveedor externo para vectorización y generación. La diferencia modifica la frontera de operación: una interrupción de embeddings puede activar recuperación léxica, mientras que una interrupción del generador impide completar la respuesta. No se comparan costos, privacidad efectiva ni calidad entre ambos sistemas, porque no se utilizaron iguales documentos, preguntas y condiciones.

El trabajo de Guerra y colaboradores [4] comparte el interés por consultas sobre PDF, pero su enrutamiento

por carrera y su dominio académico no están implementados aquí. El aporte del presente estudio se concentra en hacer observable la relación entre flujo técnico, estados de la API y evidencia de verificación. Las técnicas RRF y de reordenamiento proceden de métodos existentes [5], [6]; la integración descrita no demuestra una mejora algorítmica original.

La principal amenaza a la validez es la distancia entre los dobles de prueba y los servicios reales. Los casos aislados no ejercitan la calidad de embeddings, el planificador SQL, la extracción de documentos complejos ni la variabilidad de la generación. Tampoco prueban que una coincidencia léxica sea suficiente para contestar correctamente. La inspección encontró una base para ampliar la evaluación, no un sustituto de ella.

Otros límites son el corpus demostrativo mínimo, la ausencia de anotadores y participantes, el carácter provisional del umbral y la falta de un registro completo de recuperación. La configuración de dependencias incluye un intervalo de versiones para el SDK generativo; una réplica integral debe fijar la versión efectivamente instalada. Las huellas de archivo permiten identificar el código verificado, pero no reconstruyen el estado histórico del proveedor o de la base.

Protocolo para la siguiente fase

La siguiente fase debe comenzar con un corpus autorizado y versionado, un inventario documental y un banco de consultas con respuestas de referencia y localizadores verificados. La calibración del umbral y de los parámetros de recuperación debe realizarse en un subconjunto separado del conjunto final de evaluación. Deben incluirse preguntas sin respuesta, versiones contradictorias y consultas que fallen en la recuperación semántica o léxica.

La evaluación documental debe informar por separado recuperación, sustento de afirmaciones, exactitud de citas, abstención y latencia. Context Recall puede operacionalizarse como la proporción de afirmaciones de referencia apoyadas por el contexto recuperado, según la definición documentada por Ragas [17]. Esa métrica no debe confundirse con F1 de respuesta ni con éxito de resolución de tickets. La evaluación automática necesita controles manuales y un registro de la configuración del evaluador.

Para estudiar la utilidad operativa será necesario medir el tiempo de localización documental con analistas y tareas comparables, controlar el orden de uso y la repetición de consultas y determinar el tamaño muestral antes del contraste. El tiempo de consulta debe diferenciarse del tiempo total de resolución de una incidencia. Los objetivos de mejora de la tesis pueden orientar esa fase, pero no se consideran logros de este artículo.

Conclusiones

Se describió y verificó un prototipo RAG híbrido que integra ingesta de PDF con texto, recuperación densa y léxica, fusión RRF, selección de fuentes y generación mediante un proveedor externo. La inspección permitió delimitar con precisión los componentes existentes y los mecanismos que permanecen opcionales o pendientes.

Las 18 pruebas existentes finalizaron aprobadas y ocho escenarios adicionales caracterizaron el manejo de evidencia y de fallos. Los resultados permiten identificar controles funcionales y límites concretos: una abstención textual puede no reflejarse en el estado estructurado, las fuentes recuperadas no constituyen verificación de afirmaciones y el registro no confirma persistencia ante errores. Estas observaciones orientan mejoras específicas de implementación y evaluación.

La evidencia respalda una contribución de desarrollo y verificación de software con alcance acotado. No demuestra reducción de tiempos de atención, exactitud sobre normativa real ni superioridad frente a otras arquitecturas. La evaluación documental y operativa propuesta permitirá contrastar esas propiedades en una fase posterior, con datos y procedimientos trazables.

Disponibilidad de materiales

El material suplementario conserva el código examinado, las pruebas existentes, el procedimiento de caracterización, sus resultados y las huellas SHA-256. Las pruebas aisladas pueden ejecutarse sin credenciales de servicios externos. La demostración visual previa se conserva en la documentación del proyecto y se describe como evidencia histórica, no como un experimento reproducido en esta ejecución.

Contribución de Autoría

Moisés Alexander Félix Romero: Investigación, Metodología, Escritura – revisión y edición.

Referencias

[1] P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020. [Online]. Available: https://arxiv.org/abs/2005.11401

[2] Y. Gao *et al.*, “Retrieval-augmented generation for large language models: A survey,” arXiv:2312.10997, 2023, versión 5, 2024. [Online]. Available: https://arxiv.org/abs/2312.10997

[3] V. F. Manchay García, "Desarrollo de un prototipo de chatbot inteligente basado en arquitectura de generación aumentada por recuperación (RAG), para la automatización de consultas frecuentes en la mesa de ayuda de una empresa de servicios," tesis de grado, Universidad de Piura, 2026. [Online]. Available: https://pirhua.udep.edu.pe/backend/api/core/bitstreams/1dc4bbf2-dc5d-48c1-ae19-fbe01fa07027/content

[4] F. Guerra, K. Monti, and J. M. Fernández, "Sistema de generación aumentada por recuperación (RAG) para el acceso a información académica," in *Libro de actas del XXXI Congreso Argentino de Ciencias de la Computación*, 2025, pp. 1165–1169. [Online]. Available: https://sedici.unlp.edu.ar/handle/10915/191535

[5] G. V. Cormack, C. L. A. Clarke, and S. Büttcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," in *SIGIR '09*, 2009, pp. 758–759. [Online]. Available: https://doi.org/10.1145/1571941.1572114

[6] R. Nogueira and K. Cho, "Passage re-ranking with BERT," arXiv:1901.04085, 2019. [Online]. Available: https://arxiv.org/abs/1901.04085

[7] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "RAGAs: Automated evaluation of retrieval augmented generation," in *Proceedings of EACL 2024: System Demonstrations*, 2024, pp. 150–158. [Online]. Available: https://doi.org/10.18653/v1/2024.eacl-demo.16

[8] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004. [Online]. Available: https://doi.org/10.2307/25148625

[9] pypdf, "Extract text from a PDF," documentación técnica, consulta: 23 oct. 2026. [Online]. Available: https://pypdf.readthedocs.io/en/stable/user/extract-text.html

[10] Google, "Embeddings," Gemini API, documentación técnica, consulta: 23 oct. 2026. [Online]. Available: https://ai.google.dev/gemini-api/docs/embeddings

[11] pgvector, "Open-source vector similarity search for Postgres," repositorio y documentación técnica, consulta: 23 oct. 2026. [Online]. Available: https://github.com/pgvector/pgvector

[12] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2020. [Online]. Available: https://doi.org/10.1109/TPAMI.2018.2889473

[13] PostgreSQL Global Development Group, "Controlling text search," PostgreSQL 16 Documentation, secc. 12.3, consulta: 23 oct. 2026. [Online]. Available: https://www.postgresql.org/docs/16/textsearch-controls.html

[14] Google, "Google Gen AI SDK," documentación técnica, consulta: 23 oct. 2026. [Online]. Available: https://googleapis.github.io/python-genai/

[15] FastAPI, "Testing," documentación técnica, consulta: 23 oct. 2026. [Online]. Available: https://fastapi.tiangolo.com/tutorial/testing/

[16] pytest, "How to monkeypatch/mock modules and environments," documentación de pytest 8.3, consulta: 23 oct. 2026. [Online]. Available: https://docs.pytest.org/en/8.3.x/how-to/monkeypatch.html

[17] Ragas, "Context Recall," documentación técnica, consulta: 23 oct. 2026. [Online]. Available: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/context_recall/