Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Ingeniería en Electricidad y Computación

"DISEÑO DE UN AGENTE DE BÚSQUEDA INTELIGENTE PARA FACILITAR INFORMACIÓN DE REQUERIMIENTOS EN UNA EMPRESA DE TELECOMUNICACIONES"

Proyecto de Titulación

Previo la obtención del Título de:

MAGISTER EN SISTEMAS DE INFORMACIÓN GERENCIAL

Presentado por:

DANIELA DEL ROCÍO LINO QUIMIS

BYRON VICENTE ANTÓN ESPINOZA

Guayaquil - Ecuador
Año: 2025

Dedicatoria

El presente proyecto lo dedico, en primer lugar, a Dios, por ser mi guía constante, darme fortaleza en los momentos difíciles y acompañarme en cada paso de este camino.

A mi familia, mi pilar fundamental, por su apoyo incondicional, su amor inquebrantable y por creer siempre en mí. Me siento profundamente orgullosa de tenerlos a mi lado, pues han sido mi mayor fuente de motivación para seguir adelante.

Porque con esfuerzo, fe y perseverancia, todo es posible. Porque quien persevera, alcanza.

Daniela Lino Quimis.

En primer lugar, agradezco a Dios, por ser mi guía, mi fortaleza constante y la fuente de sabiduría que ha iluminado cada paso de este camino. Sin él, este logro no habría sido posible.

A mi querida familia, por su apoyo incondicional, sus palabras de aliento y por creer en mí incluso cuando yo dudaba. Gracias por ser mi inspiración diaria y el motor que me impulsa a seguir adelante.

Este trabajo es para ustedes, con todo mi corazón

Byron Antón Espinoza.

Agradecimientos

Mi más sincero agradecimiento a los profesores por su apoyo y por compartir sus conocimientos a lo largo de esta etapa.

Agradezco también a mi empresa por brindarme la oportunidad y las facilidades necesarias para llevar a cabo este trabajo.

Daniela Lino Quimis

A mis docentes **y** tutores, por compartir su conocimiento y orientarme con sabiduría y compromiso. Su guía académica ha sido clave en la construcción de esta tesis y en mi crecimiento profesional.

A mis compañeros de maestría, por su compañía, sus consejos y por ser parte de esta etapa. Compartir este recorrido con ustedes lo hizo más llevadero y significativo.

Byron Antón Espinoza.

Declaración Expresa

Nosotros Daniela del Rocío Lino Quimis, Byron Vicente Antón Espinoza acordamos y reconocemos que:

La titularidad de los derechos patrimoniales de autor (derechos de autor) del proyecto de graduación corresponderá al autor o autores, sin perjuicio de lo cual la ESPOL recibe en este acto una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra con facultad de sublicenciar, incluyendo la autorización para su divulgación, así como para la creación y uso de obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores.

La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada por mí/nosotros durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que me/nos corresponda de los beneficios económicos que la ESPOL reciba por la explotación de mi/nuestra innovación, de ser el caso.

En los casos donde la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique al/los autor/es que existe una innovación potencialmente patentable sobre los resultados del proyecto de graduación, no se realizará publicación o divulgación alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 13 de Agosto del 2025.

Daniela Del Rocio Lino Quimis
Time Stamping Security Data

Ing. Daniela del Rocío Lino Quimis

Byron Vicente Anton Espinoza
Time Stamping Security Data

Ing Byron Vicente Antón Espinoza

Evaluadores

Firmado electrónicamente por:
JUAN CARLOS GARCIA PLUA
Validar únicamente con FirmaEC

Ph.D. Juan Carlos García Plúa

Tutor de Proyecto

Firmado electrónicamente por:
LENIN EDUARDO FREIRE COBO
Validar únicamente con FirmaEC

Mgtr. Lenín Freire Cobo
Revisor de Proyecto

RESUMEN

Este proyecto propone el diseño de un agente inteligente orientado a facilitar el acceso a información técnica sobre requerimientos funcionales en una empresa de telecomunicaciones. Se plantea como objetivo principal mejorar la forma en que los usuarios consultan documentación histórica, reduciendo ambigüedad, tiempo de búsqueda y pérdida de información. La propuesta se justifica en la necesidad de automatizar procesos de recuperación de información y aprovechar tecnologías de inteligencia artificial aplicadas al contexto empresarial. Para el desarrollo del sistema, se utilizaron herramientas de código abierto como GitLab, Qdrant, n8n y el modelo de lenguaje LlamA. Se diseñó una arquitectura modular tipo RAG (Retrieval-Augmented Generation), que integró flujos automatizados, vectorización de textos técnicos y generación de respuestas en lenguaje natural. Los resultados demostraron que la solución es técnicamente viable, funcional en entornos locales y capaz de responder con precisión a preguntas formuladas en lenguaje natural. Asimismo, se comprobó la utilidad de aplicar IA generativa sobre datos técnicos estructurados. Se concluye que el agente inteligente propuesto puede adaptarse a la infraestructura actual de la empresa, optimizando la consulta de requerimientos y fortaleciendo la reutilización del conocimiento técnico.

ABSTRACT

This project proposes the design of an intelligent agent aimed at facilitating access to technical information on functional requirements within a telecommunications company. The main objective is to improve how users retrieve historical documentation, reducing ambiguity, search time, and loss of relevant information. The proposal is justified by the need to automate information retrieval processes and to apply artificial intelligence technologies in enterprise environments. For the development of the system, open-source tools such as GitLab, Qdrant, n8n, and the Llama language model were used. A modular RAG-based (Retrieval-Augmented Generation) architecture was designed, integrating automated workflows, technical text vectorization, and natural language response generation. The results showed that the solution is technically feasible, functional in local environments, and capable of responding accurately to natural language questions. Additionally, the usefulness of applying generative AI to structured technical data was confirmed. It is concluded that the proposed intelligent agent can be integrated into the company's current infrastructure, optimizing requirements consultation and enhancing the reuse of technical knowledge.

Índice general

Abreviaturas

APIs (Application Programming Interfaces)

BPMN Business Process Model and Notation

LLM Large Language Models

ESPOL Escuela Superior Politécnica del Litoral

NLP — Natural Language Processing (Procesamiento del Lenguaje Natural).

LLaMA — Large Language Model Meta AI

PDF — Portable Document Format

HTTP — Hypertext Transfer Protocol

RAG Retrieval-Augmented Generation

Simbología

Índice de figuras

Índice de tablas

CAPÍTULO 1

1. INTRODUCCIÓN

1.1 Descripción del Problema

Dentro del contexto de los sistemas de información y del sector de telecomunicaciones, la correcta elaboración de los requerimientos funcionales es un aspecto determinante para garantizar la eficacia y viabilidad de los proyectos tecnológicos.

En una empresa de telecomunicaciones de gran tamaño con múltiples departamentos, objeto de este estudio, se han identificado dificultades durante el proceso de elaboración de requerimientos las cuales afectan la calidad de las soluciones planteadas.

Como consecuencia, se producen retrasos en el ingreso oportuno de los requerimientos y se generan múltiples iteraciones de reuniones para el levantamiento inicial, necesarias para aclarar, corregir y redefinir continuamente las especificaciones funcionales. Esta situación no solo impacta en los tiempos y recursos de los proyectos, sino que también limita la claridad en la definición de las verdaderas necesidades del negocio y su correcta traducción a soluciones técnicas viables.

La construcción de este buscador inteligente es uno de los primeros proyectos que servirá de base para futuras implementaciones con inteligencia artificial en la empresa. Este diseño busca fomentar la adaptación de esta tecnología en nuevos proyectos, permitiendo que en etapas posteriores pueda ser considerado como un componente dentro de un ecosistema más amplio de soluciones basadas en IA.

El presente trabajo de titulación tiene como propósito el diseño de un asistente inteligente de búsqueda que estará orientado a facilitar el acceso a información histórica sobre requerimientos, permitiendo a los usuarios consultar ejemplos previos y obtener referencias útiles que les ayuden a redactar nuevos requerimientos de forma más clara. El diseño propuesto busca que el usuario pueda reducir tiempos al crear sus requerimientos y evitar la ambigüedad del mismo, minimizar la necesidad de iteraciones innecesarias y mejorar la alineación entre las necesidades del negocio.

En las empresas de telecomunicaciones, caracterizada por una alta complejidad técnica y la participación de múltiples áreas, la falta de un sistema estructurado para recuperar información de proyectos anteriores representa un desafío constante. Actualmente, consultar antecedentes sobre requerimientos implementados suele implicar recurrir a correos antiguos o realizar consultas informales, lo que genera pérdida de tiempo y retrabajo. En muchos casos, aunque se sepa de una funcionalidad que ya fue desarrollada, no se tiene claridad sobre a qué proyecto perteneció, quién fue su responsable o bajo qué requerimiento se ejecutó.

1.2 Justificación del Problema

El diseño de un sistema de búsqueda inteligente, basado en consultas en lenguaje natural e integrado con un repositorio reorganizado como GitLab, responde a la necesidad de optimizar la búsqueda y recuperación de información útil para el diseño de nuevos requerimientos. Este enfoque facilitará un acceso rápido a la información para los usuarios, al reutilizar los datos existentes de los proyectos.

Además, el sistema se apoya en principios de exploración de datos, lo que le permite adaptar sus respuestas en función de las necesidades del usuario y de los datos almacenados en el repositorio. Esta capacidad incrementa su utilidad a medida que se enriquece el repositorio y se afina la interacción. En consecuencia, la propuesta no solo ofrece un apoyo efectivo en la redacción de requerimientos desde las etapas iniciales del proyecto, sino que también establece las bases para futuras implementaciones que integren inteligencia artificial aplicada a la toma de decisiones, fortaleciendo así la eficiencia y trazabilidad en la gestión de proyectos tecnológicos.

Esta técnica se utiliza en diversas áreas y ha demostrado ser efectiva, ofreciendo soluciones flexibles y seguras para abordar una variedad de problemas [1] . El estudio de [2] aborda los desafíos y avances en la implementación de modelos de lenguaje grandes (LLMs), que pueden ser fundamentales para el análisis de requerimientos**.** También las ventajas de utilizar programas para el tratamiento automático de datos puede facilitar la validación de requisitos [3] .

Adicional los sistemas de gestión de flujos de trabajo deben transformar los procesos de negocio en expresiones comprensibles por las computadoras, lo que es esencial para la automatización en este contexto [4]. En conclusión en el estudio de [5] resaltan el potencial transformador de la Inteligencia Artificial (IA) para optimizar las tareas de gestión de proyectos.
Este proyecto es viable, ya que se cuenta con acceso a los actores principales con quienes se podrá interactuar para recopilar información sobre problemas recurrentes en el proceso de gestión de requerimientos. Además, el alcance del proyecto se enfoca en desarrollar un diseño estructurado que sirva como base para la implementación de un buscador inteligente. Asimismo, se dispone de acceso a la información histórica y actual de requerimientos, lo cual permite realizar un análisis adecuado y garantizar la calidad del diseño propuesto.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar un sistema de búsqueda inteligente, que apoye a los usuarios en el levantamiento de requerimientos funcionales, facilitando el acceso a la información mediante la reutilización de información histórica de proyectos anteriores, integrando modelos de lenguaje natural y flujos automatizados con acceso al repositorio GitLab.

1.3.2 Objetivos específicos

1. Identificar los puntos de dolor en la búsqueda y recuperación de requerimientos funcionales, en los equipos de negocio y técnicos mediante formularios estructurados que permitan evidenciar ambigüedades, tiempos de respuestas y pérdida de información útil.
2. Estandarizar la forma en la que se registra la información de los proyectos en el repositorio de GitLab, organizando los requerimientos de manera lógica y accesible para facilitar su búsqueda y consulta en el análisis de datos históricos.
3. Definir los flujos de interacción y arquitectura de la búsqueda inteligente, mediante prototipos de interfaz que conecte el modelo de lenguaje natural (LLM) con información extraída de GitLab mediante flujo automatizado N8N, permitiendo de esta forma interpretar consultas en lenguaje natural y entregar respuestas contextualizadas a usuarios.

1.4 Marco teórico

El presente capítulo tiene como propósito presentar los fundamentos teóricos y tecnológicos que respaldan el desarrollo del agente inteligente de búsqueda propuesto. Para esto, se explorarán las distintas herramientas y metodologías que permiten estructurar soluciones orientadas a optimizar el levantamiento de requerimientos en entornos complejos como el de una empresa de telecomunicaciones. Se abordarán temas como la automatización de procesos mediante flujos orquestados, la organización eficiente de repositorios en plataformas colaborativas como GitLab, y la aplicación de modelos avanzados de lenguaje natural para interpretar consultas formuladas por los usuarios. Además, se revisarán las características claves relacionados con los agentes inteligentes, su evolución histórica y su contribución a la optimización de procesos dentro del ámbito de la ingeniería de software. Esta revisión permitirá comprender los elementos que sustentan el diseño propuesto y establecerá los criterios técnicos sobre los cuales se construirá una solución innovadora, flexible y alineada con los desafíos actuales de la gestión de proyectos tecnológicos.

1.4.1 Modelado del flujo automatizado

Durante los últimos años la automatización de procesos, se ha convertido en un elemento central dentro de la transformación digital de las organizaciones, el uso adecuado de los recursos en la operativa es uno de los factores claves para la mejora del rendimiento, al facilitar la disminución de errores humanos y acelerar la capacidad de respuesta, uno de los modelados formal que se tiene es el caso de Business Process Model and Notation (BPMN 2.0), donde se puede mostrar los flujos actuales y esperados mediante grafico de actividades, decisiones y eventos, en este punto permite no solo documentar, sino ejecutar flujos operativos de forma estructurada [6] a través de técnicas de análisis y optimización formal [7].Plataformas low-code como n8n permiten aplicar este enfoque en entornos visuales e interactivos, facilitando la construcción de secuencias de procesos automatizados mediante nodos que reaccionan a eventos como llamadas API, transformaciones o disparadores externos, sin requerir programación intensiva [8] lo cual permite una adaptabilidad de los cambios del negocio, todo esto indica que un flujo automatizado bien diseñado parte de una definición precisa de las actividades y sus interrelaciones, lo que permite organizar los pasos en una secuencia lógica y ofrecer una visualización clara del sistema. Esta claridad no solo facilita la colaboración entre perfiles técnicos y no técnicos, sino que, al integrarse con nuevas estrategias de automatización, mejora significativamente la interacción entre los procesos automáticos y los usuarios humanos [9].
GitLab en la actualidad para el desarrollo colaborativo de software se ha consolidado una plataforma integral, dentro del ciclo completo del desarrollo ya que abarca desde la planificación de un proyecto monitoreo del desarrollo, y hasta

la mejora continua (CI/CD). La arquitectura DevOps permite colocar dentro de la herramienta tareas para control de versiones con Git, monitoreo y gestión de incidencias, la documentación en Markdown, las pruebas automatizadas y la integración continua. Toda esta centralización dentro de la herramienta permite tener una mayor trazabilidad y colaboración entre equipos técnicos y no técnicos, facilitando la transparencia y el versionamiento de los artefactos producidos.

Para el contexto de los requerimientos de software con respecto a proyectos, lo convierte en un repositorio robusto por su sistema de gestión de issues/incidencias, milestones, y documentación. Aquí se almacena información relevante de proyectos y permite toma de decisiones, se podrá evidenciar los cambios dentro de las funcionalidades y reglas que tiene el negocio. Por todas sus ventajas en datos históricos, y para el análisis retrospectivo de requerimientos y la reutilización del conocimiento almacenado, convierte a la herramienta en un punto clave.

Esta herramienta ofrece comunicaciones con otras plataformas a través de API REST, que por medio de parámetros permite acceder a la información almacenada, lo cual facilita la integración con procesos automatizados, para alimentar la información del agente que sea capaz de extraer, analizar y contextualizar información sobre requerimientos anteriores para facilitar la toma de decisiones en etapas tempranas del ciclo de vida del software.

En investigaciones recientes, se ha demostrado que los repositorios de software no solo son herramientas de gestión operativa, sino también fuentes útiles para la minería de procesos orientada a la mejora continua. Por ejemplo,[10] analizan cómo los repositorios de desarrollo pueden alimentar retrospectivas ágiles y revelar patrones de colaboración, comunicación y evolución técnica dentro de equipos de ingeniería de software. Esta evidencia empírica valida el uso de plataformas colaborativas como GitLab como base para la construcción de sistemas inteligentes que apoyen tareas complejas como la recuperación de requerimientos y el diseño asistido de soluciones.

En el diseño propuesto, el agente de búsqueda se basa en la ejecución de un flujo automatizado que extrae información desde archivos estructurados (DERCAS) adjuntos a issues en GitLab. Esto permite alimentar al agente con datos de forma constante y controlada. A continuación, se presentan las funcionalidades clave de GitLab que hacen posible esta arquitectura de extracción automatizada.

Tabla I. Funcionalidades Gitlab vs Agente

Funcionalidad de GitLab	**Aplicación en la solución propuesta**	**Beneficio para el agente inteligente**
Seguimiento de issue	Almacena solicitudes técnicas donde se adjuntan archivos DERCAS.	Permite al agente localizar requerimientos de la herramienta.
Archivos adjuntos en issues	Se carga la información en un issue	Permite una recolección automatizada de datos sin necesidad de explorar todo el repositorio.
API RESTful	Obtiene información como nombres de archivo, autores, fechas y otros metadatos vinculados a un *issue*.	Clasificación por autor o por tipo de requerimiento.
Historial de cambios (Git)	Permite la carga de los archivos y sus versiones	Permite al agente contextualizar cómo ha evolucionado un requerimiento documentado

En la actualidad, los flujos automatizados han revolucionado la forma en que las organizaciones llevan a cabo sus operaciones, ofreciendo una gran eficiencia y una capacidad de respuesta ágil ante las demandas del mercado. La **retroalimentación inmediata** se ha convertido en un elemento fundamental de esta transformación, mejorando notablemente la toma de decisiones y la gestión de procesos.

Los flujos automatizados permiten que las tareas repetitivas y monótonas se realicen sin necesidad de intervención humana, lo que libera a los empleados para que se enfoquen en actividades más estratégicas. Por ejemplo, en una empresa de comercio electrónico, un flujo automatizado puede encargarse del procesamiento de pedidos, desde la recepción hasta la confirmación y el envío. Esto no solo acelera el proceso, sino que también minimiza la posibilidad de errores humanos, resultando en una experiencia más fluida para el cliente.

La retroalimentación inmediata se integra en estos flujos automatizados mediante sistemas de monitoreo y análisis de datos. Cuando un cliente realiza un pedido, el sistema puede enviar automáticamente una notificación al equipo de ventas y al departamento de logística. Si se presenta un retraso en el envío, el sistema puede alertar a los responsables, quienes pueden tomar medidas de manera inmediata para solucionar el inconveniente. Esta capacidad de respuesta

rápida es esencial en un entorno empresarial donde la satisfacción del cliente es crucial.
En los últimos años el auge de la IA conversacional y los chatbots, como ChatGPT, ha comenzado a revolucionar varios sectores. La capacidad de recibir información instantánea permite a estos agentes ajustar sus respuestas y recomendaciones en función de las interacciones del usuario [11].
La automatización del flujo de recuperación de información a través de un agente de búsqueda inteligente transforma el **modelo de procesos** to-be al sustituir tareas manuales de exploración documental por una extracción estructurada y automática desde archivos subidos a plataformas como GitLab. Esta incorporación no altera el contenido de los requerimientos, pero sí agiliza su disponibilidad y trazabilidad, optimizando los tiempos de consulta en entornos donde la documentación técnica es extensa. Este tipo de automatización especializada responde al enfoque de Intelligent Process Automation centrado en tareas cognitivas delimitadas, como la clasificación, lectura y vinculación de archivos técnicos[12], [13].
Ha sido un proceso continuo desde hace mucho tiempo atrás y ha estado orientado por avances en inteligencia artificial y computación, tales como: Sistemas Basados en Conocimiento, Agentes autónomos, Procesamiento del Lenguaje Natural, Modelos de Lenguaje. Reflejan un avance constante en la comprensión y aplicación de la inteligencia artificial en los agentes inteligentes.

1.4.2 Agentes Inteligentes

Dentro de la evolución de los **Agentes Inteligentes** los principales fueron los Sistemas Basados en Conocimiento (Knowledge-Based Systems), que operaban mediante reglas explícitas codificadas por expertos para simular procesos de toma de decisiones en dominios específicos. Aunque efectivos en un entorno o área muy limitada, estos sistemas carecían de flexibilidad para adaptarse a nuevos escenarios y dependían completamente del conocimiento explícito programado.
Posteriormente, el desarrollo de agentes autónomos supuso un avance significativo: estos agentes podían percibir su entorno, procesar información y actuar de forma independiente para alcanzar objetivos. Este paradigma se amplió con los sistemas multiagente, donde múltiples agentes interactúan y cooperan para resolver problemas complejos y distribuidos, como ocurre en la gestión dinámica del tráfico de datos en redes de telecomunicaciones o la coordinación en redes de sensores inalámbricos.
La incorporación del Procesamiento del Lenguaje Natural (PLN) permitió que los agentes inteligentes entendieran y generaran lenguaje humano, facilitando así la interacción natural con los usuarios. Esto propició la creación de agentes conversacionales y asistentes virtuales, como Siri, Alexa y Google Assistant, que integran tecnologías de reconocimiento de voz y análisis semántico para responder preguntas, ejecutar comandos y aprender de las interacciones.
Lo más reciente la evolución hacia modelos de lenguaje basados en aprendizaje profundo, como la arquitectura de red neuronal, que ha revolucionado los agentes inteligentes. Estos modelos, entrenados con grandes volúmenes de datos, han ampliado la capacidad de comprensión, generación de texto y razonamiento contextual de los agentes, permitiendo aplicaciones avanzadas en análisis de requerimientos, generación de documentación automática y soporte en la toma de decisiones.
La evolución de la inteligencia artificial generativa ha tenido un cambio significativo en la remodelación del futuro de la tecnología en diferentes aspectos. Las redes inalámbricas, en particular, con el crecimiento de las redes autoevolutivas, representan un gran área para explotar y cosechar varios beneficios que pueden cambiar fundamentalmente la forma en que se diseñan y operan las redes inalámbricas en la actualidad [14].
Dentro de los **beneficios** podemos encontrar que son impulsados por tecnologías de inteligencia artificial y aprendizaje automático, representan una herramienta clave en la transformación de los procesos de toma de decisiones en empresas de tecnología. Estos agentes son capaces de analizar grandes volúmenes de datos, identificar patrones complejos y generar recomendaciones o tomar decisiones autónomas en tiempo real. Entre sus principales beneficios destacan la mejora en la precisión de las decisiones, la reducción del error humano, la capacidad para operar a gran escala y velocidad, y la automatización de tareas repetitivas. Además, permiten personalizar respuestas según el contexto y anticipar comportamientos futuros mediante modelos predictivos.
La innovación continua es otra ventaja, ya que los agentes inteligentes pueden aprender y adaptarse a nuevas situaciones a través de técnicas de aprendizaje automático. Esto significa que su rendimiento puede mejorar con el tiempo, permitiendo a las organizaciones mantenerse competitivas en un entorno en constante cambio.
La escalabilidad es otro aspecto destacado. Los agentes inteligentes pueden adaptarse a diferentes volúmenes de trabajo sin necesidad de aumentar proporcionalmente los recursos humanos. Esto permite a las empresas crecer y expandirse sin enfrentar los mismos desafíos que tendrían al aumentar su personal.
Su aplicación se ha demostrado efectiva en diversos sectores como las telecomunicaciones, salud, finanzas y logística, facilitando desde diagnósticos médicos y detección de fraudes hasta optimización de redes y procesos operativos. En conclusión, la implementación de agentes inteligentes no solo potencia la eficiencia operativa, sino que también impulsa la innovación estratégica al transformar datos complejos en conocimientos accionables [15].
La definición del agente es considerada como un sistema autónomo capaz de percibir, razonar y actuar para alcanzar

objetivos específicos. Se clasifican en diversos tipos como agentes reactivos, deliberativos o de aprendizaje, cada uno con aplicaciones adecuadas para el análisis y validación automatizada de requisitos [16]
Los agentes reactivos son sistemas que responden a estímulos del entorno de manera inmediata, sin mantener un modelo interno del estado del mundo. Su comportamiento se basa en reglas simples de condición y acción, lo que les permite actuar rápidamente ante cambios en su entorno.
Los agentes deliberativos son sistemas que poseen un modelo interno del entorno y utilizan este modelo para razonar y planificar sus acciones. Estos agentes pueden prevenir las consecuencias de sus decisiones y, por lo tanto, son capaces de tomar decisiones más informadas y estratégicas a largo plazo.
Los agentes de aprendizaje son sistemas que tienen la capacidad de mejorar su rendimiento a lo largo del tiempo mediante la experiencia. Utilizan técnicas de aprendizaje automático para adaptarse a cambios en el entorno y optimizar su comportamiento, aprendiendo de sus errores y ajustando sus estrategias en función de la retroalimentación recibida.
La elección del tipo de agente inteligente depende del contexto específico y los objetivos de la automatización. En particular, para el análisis y validación automatizada de requisitos, se requieren agentes que combinen capacidades de razonamiento estructurado con aprendizaje adaptativo para manejar la complejidad y variabilidad inherente a los proyectos de desarrollo de software.
Existe una creciente demanda de agentes inteligentes en sectores tecnológicos para automatizar procesos complejos, entre ellos la validación de requerimientos, impulsada por la necesidad de optimizar tiempos y reducir errores en proyectos de software de gran escala. Los avances recientes prometen superar las barreras existentes al mejorar el procesamiento del lenguaje natural y las capacidades de razonamiento. Si bien es prometedor, para crear asistentes virtuales más avanzados aún enfrenta desafíos como garantizar un rendimiento robusto y gestionar la variabilidad de los comandos de usuario en el mundo real.
Además, la personalización de servicios se ha convertido en una prioridad. Las empresas están utilizando agentes inteligentes para ofrecer experiencias adaptadas a las preferencias individuales de los usuarios, lo que no solo mejora la satisfacción del cliente, sino que también fomenta la lealtad a la marca. Los agentes son capaces de analizar datos en tiempo real, lo que les permite anticipar necesidades y ofrecer recomendaciones personalizadas.
Una tendencia importante es el desarrollo de agentes autónomos, que pueden operar de manera independiente. Esto permite a las organizaciones delegar tareas críticas sin necesidad de intervención humana, aumentando la eficiencia y reduciendo la posibilidad de errores. Además, los agentes inteligentes están diseñados para colaborar con los humanos, complementando sus habilidades y facilitando una cooperación más efectiva en el entorno laboral.
El artículo de [17] propone un novedoso asistente virtual basado en LLM que puede realizar automáticamente operaciones de varios pasos dentro de aplicaciones móviles, basándose en solicitudes de usuario de alto nivel.

Las bases de datos vectoriales se han convertido en una pieza importante para los sistemas que necesitan buscar información a partir de una base de conocimiento y no solo por coincidencia exacta de palabras. Este tipo de tecnología guarda representaciones numéricas de los datos, llamadas vectores o embeddings, que conservan relaciones de significado y permiten que una consulta devuelva resultados que están relacionados conceptualmente aunque usen un vocabulario diferente[18]. Para encontrar rápidamente elementos similares, utilizan algoritmos como HNSW (Hierarchical Navigable Small World) , reconocidos por su velocidad y precisión en espacios de alta dimensión [19] [20]. En este ecosistema, Qdrant destaca como una solución open-source optimizada para búsquedas vectoriales, integrando HNSW con filtrado avanzado, ideal para arquitecturas RAG donde un modelo de lenguaje aprovecha el contexto recuperado para generar respuestas más relevantes [21] .
En el diseño de esta investigación, Qdrant actúa como repositorio semántico de requerimientos funcionales. Su conexión con GitLab, N8N y el modelo LLaMA se hace mediante integración punto a punto: un canal directo que favorece la baja latencia y simplicidad. Aunque este enfoque crea una interdependencia entre componentes, en este prototipo controlado es efectivo porque simplifica la implementación y asegura que las respuestas estén sustentadas en datos técnicos revisados.

1.4.3 Modelos de Lenguaje Grandes (LLMs).

Se define a los modelos de lenguaje Grande como sistemas de inteligencia artificial diseñados para comprender, generar y manipular el lenguaje humano de manera efectiva, entrenados con volúmenes masivos de texto. Su arquitectura se basa en la estructura de transformadores, que permite procesar secuencias de datos de manera eficiente y capturar relaciones contextuales a gran escala. Se destacan por su capacidad para aprender patrones complejos en el lenguaje, lo que les permite realizar diversas tareas, como traducción automática, generación de texto, respuesta a preguntas y asistencia en procesos de toma de decisiones. Esto mejora significativamente la interacción entre los sistemas y los usuarios en aplicaciones de procesamiento de lenguaje natural.
En el artículo de [21] presenta una herramienta de diseño de arquitectura basada en el modelo de lenguaje grande, que logra un diseño de arquitectura inteligente mediante la comprensión de las intenciones del usuario y la generación y optimización de diseños. La herramienta de diseño de arquitectura es responsable de recibir las entradas de diseño del

usuario y mostrar los resultados del diseño, mientras que el modelo de lenguaje grande se encarga de la tarea de diseño de arquitectura inteligente.

Dentro de los tipos que han transformado de manera significativa el ámbito del procesamiento del lenguaje natural (NLP), abriendo la puerta a una amplia gama de aplicaciones que incluyen desde la generación de texto hasta la traducción automática. A continuación, se describen los principales tipos de LLM, cada uno con sus propias características y aplicaciones específicas:

Modelos Generativos, diseñados para generar texto coherente y relevante a partir de una entrada dada. Utilizan técnicas de aprendizaje profundo para predecir la siguiente palabra en una secuencia, lo que les permite crear contenido original. Como referencia tenemos a GPT-3 y GPT-4, que han demostrado capacidades avanzadas en la generación de texto.

Modelos de Traducción, Especializados en traducir texto de un idioma a otro, estos modelos utilizan LLM para mejorar la precisión y fluidez de las traducciones. Un ejemplo notable es Google Translate, que ha integrado LLM para ofrecer traducciones más contextuales y precisas.

Modelos de Resumen, Estos modelos son capaces de condensar información extensa en resúmenes breves y concisos, extrayendo los puntos clave de documentos largos. Se utilizan en aplicaciones que requieren la síntesis de información, como la generación de resúmenes de artículos o informes.

Modelos de Conversación, Diseñados para interactuar con los usuarios en un formato de diálogo, estos modelos permiten mantener conversaciones naturales y fluidas. Los chatbots que utilizan LLMs son un ejemplo de esta categoría, brindando asistencia y respuestas a preguntas en tiempo real.

Modelos Multimodales, Estos modelos son capaces de procesar y generar no solo texto, sino también imágenes y otros tipos de datos. Un ejemplo es Gemini de Google, que combina texto e imágenes para ofrecer respuestas más completas y contextuales.

Modelos de Código, Especializados en la generación y comprensión de código de programación, estos modelos ayudan a los desarrolladores a escribir código de manera más eficiente. OpenAI Codex es un ejemplo que permite a los programadores interactuar con el modelo para obtener sugerencias y soluciones de codificación.

En conclusión, los LLM son una parte esencial del procesamiento informático del lenguaje, ya que permiten comprender patrones verbales complejos y generar respuestas coherentes y apropiadas en un contexto determinado [22].

A continuación, se detalla un cuadro con los tipos de modelos más concurrentes.

Tabla II. Tipos de Modelos

Tipo de LLM	Ejemplo	Características
Modelos Generativos	GPT-3, ChatRWKV, Llama	Generan texto coherente y contextual, ideales para interacciones conversacionales.
Modelos de Conversación	Lamda, Meena	Diseñados para interactuar con los usuarios en un formato de diálogo.
Modelos de Resumen	XLNet, ALBERT	Diseñados para responder preguntas específicas basadas en un contexto dado.
Modelos de Traducción	Gemma, Alma,	Especializados en traducir texto entre diferentes idiomas.
Modelos de Multimodales	GPT-4, DALL-E	Son capaces de procesar y generar no solo texto, sino también imágenes y otros tipos de datos
Modelos de código	Copilot, DeepCode	Especializados en la generación y comprensión de código de programación

Con respecto a los **costos** asociados al desarrollo y despliegue de LLM ha experimentado una evolución significativa desde los primeros sistemas de procesamiento del lenguaje natural hasta los modelos actuales, impulsada por avances tecnológicos y cambios en la escala y complejidad de los modelos.

En las primeras etapas del procesamiento del lenguaje natural, durante las décadas de 1980 y 1990, los modelos se basaban principalmente en reglas manuales y sistemas estadísticos simples. Estos sistemas, aunque limitados en capacidad, eran relativamente económicos en términos computacionales porque operaban sobre conjuntos de datos reducidos y modelos menos complejos. Sin embargo, su alcance y precisión eran limitados, lo que obligaba a intervenciones humanas frecuentes y extensos esfuerzos en la elaboración y mantenimiento de reglas, lo que traducía en costos humanos y de tiempo elevados, aunque con baja inversión en infraestructura computacional.

Con la llegada de métodos estadísticos y de aprendizaje automático en los 2000, el tamaño de los modelos y la cantidad de datos utilizados comenzaron a aumentar considerablemente. Modelos como los basados en n-gramas o modelos de Markov ocultos demandaban mayor poder computacional para entrenarse, y se incrementaron los costos de procesamiento y almacenamiento. A medida que los modelos aumentaban en complejidad y tamaño, también lo hacían los costos asociados a la infraestructura tecnológica necesaria para su entrenamiento, almacenaje y despliegue. Las empresas e instituciones comenzaron a invertir en clusters de computación especializados y almacenamiento en grandes volúmenes.

En la actualidad, con la aparición de modelos como GPT-3, con 175 mil millones de parámetros, el costo de entrenamiento se ha disparado a niveles sin precedentes [23]. Además, los costos operativos siguen siendo altos debido a la complejidad de la inferencia, que demanda recursos potentes para brindar respuestas rápidas en aplicaciones en tiempo real.

A pesar de los altos costos actuales, se observan tendencias que apuntan a reducirlos, como el uso de hardware más eficiente, modelos más compactos mediante aprendizaje y estrategias para reutilizar modelos preentrenados. Estas innovaciones buscan popularizar el acceso a LLMs, permitiendo su uso en empresas y sectores con menores recursos. El artículo de [24] explora la incorporación de soluciones LLM como servicio en los flujos de trabajo empresariales, centrándose en los costos de inferencia. Analizamos diversas ofertas de LLM y realizan un análisis comparativo basado en un caso práctico real de un chatbot de IA.

Se ha considerado como limitaciones y consideraciones de los Modelos de Lenguaje de Grande que han revolucionado la forma en que interactuamos con la tecnología, pero también presentan varios puntos que es importante tener en cuenta. Una de las principales preocupaciones es el sesgo en los datos. Estos modelos se entrenan utilizando grandes volúmenes de información, y si esos datos contienen sesgos, el modelo puede reproducir y amplificar esos mismos sesgos en sus respuestas. Esto puede llevar a resultados injustos o inexactos.

Otro aspecto a considerar es la dependencia de los datos de entrenamiento. La calidad y diversidad de estos datos son cruciales para el rendimiento del modelo. Si los datos son limitados o no representan adecuadamente la realidad, el modelo puede tener dificultades para generalizar y ofrecer respuestas precisas. Además, entrenar y ejecutar estos modelos requiere recursos computacionales significativos, lo que puede ser un obstáculo para muchas organizaciones. Además, aunque los LLM son capaces de generar texto que parezca coherente y relevante, no poseen una comprensión profunda del contenido. Su funcionamiento se basa en patrones estadísticos, lo que significa que no entienden realmente lo que están diciendo. Esto puede resultar en la generación de información incorrecta o engañosa, ya que no tienen la capacidad de verificar hechos o acceder a información actualizada.

A pesar de su utilidad, los LLMs presentan limitaciones como la generación ocasional de respuestas incorrectas o ambiguas, dependencia de grandes cantidades de datos y consideraciones a la privacidad, factores que deben controlarse al implementar agentes inteligentes para validación de requerimientos [25].

1.4.4 casos similares de Integraciones

Entre los casos similares de integración donde los entornos de la automatización de procesos requieren respuestas dinámicas y contextualizadas, los modelos de lenguaje de gran escala (LLMs) han demostrado ser una solución eficaz al integrarse como componentes interpretativos dentro de arquitecturas controladas por eventos. Un ejemplo destacado es el trabajo desarrollado por [26], quienes diseñaron un sistema de automatización industrial basado en LLMs capaz de recibir entradas en tiempo real provenientes de sensores o eventos del entorno operativo. A través del análisis semántico de estos eventos, el modelo generaba planes de acción detallados que eran traducidos en instrucciones ejecutables por microservicios a través de APIs. Esta arquitectura no solo permitió automatizar tareas que tradicionalmente dependían de reglas codificadas, sino que introdujo flexibilidad semántica y adaptabilidad contextual en la toma de decisiones operativas. El enfoque modular del sistema, donde los LLMs funcionaban como intermediarios inteligentes entre el entorno y los servicios de control, demuestra una clara analogía con herramientas como n8n, en las que flujos visuales automatizados pueden incorporar modelos de lenguaje para interpretar entradas textuales, consultar servicios externos y desencadenar acciones de forma programada. En el contexto de esta investigación, dicho paradigma resulta aplicable a la extracción de información desde archivos técnicos como los DERCAS en GitLab, permitiendo que el agente de búsqueda estructure respuestas automáticas basadas en el contenido recibido, sin intervención humana directa. La validación de este enfoque en un entorno de automatización industrial con alta demanda de precisión y sincronización refuerza la solidez del modelo propuesto en esta tesis para el ámbito de telecomunicaciones.

En la investigación llevada a cabo por [27] se evaluaron diversos modelos de lenguaje de gran escala (LLMs) aplicados a tareas automatizadas relacionadas con el procesamiento del lenguaje natural. El análisis incluyó indicadores clave como la exactitud, la completitud de la información, la pertinencia de las respuestas, su claridad, y el tiempo requerido para generarlas. Estos resultados ofrecen un punto de referencia cuantitativo sobre el rendimiento de los LLMs en contextos de automatización semántica, lo cual resulta pertinente para la solución propuesta en esta tesis, donde se busca estructurar información técnica mediante un agente de búsqueda inteligente. La Tabla resume los valores obtenidos en dicha evaluación.

Tabla III. Precisión, Completitud, Relevancia, Comprensibilidad, Legibilidad y Tiempo de Respuesta promediados entre todos los modelos de lenguaje (LLMs) [27]

Métrica	Precisión	Integridad	Relevancia	Comprensibilidad	Legibilidad	Tiempo Tomado
Media	2.860	4.192	3.833	3.551	4.525	16.977
Desviación estándar	0.445	1.206	0.567	0.766	0.935	17.994
Mínimo	0.000	0.000	0.000	0.000	0.000	0.000

25% (Q1)	3.000	4.000	4.000	3.000	4.000	0.000
50% (Mediana)	3.000	5.000	4.000	4.000	5.000	11.000
75% (Q3)	3.000	5.000	4.000	4.000	5.000	30.250
Máximo	3.000	5.000	4.000	4.000	5.000	75.000

Diversas **investigaciones recientes** sobre casos de LLm en entornos colaborativos han abordado la incorporación de inteligencia artificial, especialmente modelos de lenguaje de gran escala (LLMs), en entornos de desarrollo colaborativo como GitHub y GitLab, con el objetivo de automatizar la recuperación, organización y generación de información técnica. Estos casos permiten evidenciar cómo otras soluciones han enfrentado desafíos similares, como el acceso estructurado a requerimientos, el análisis de incidencias y la documentación automatizada.

En el estudio [28] se evidencia el uso de modelos de lenguaje en proyectos públicos de GitHub, demostrando que herramientas como Copilot y ChatGPT son aplicadas para generar fragmentos de código, automatizar transformaciones de datos y redactar documentación técnica en tiempo real. Esta evidencia respalda la viabilidad de utilizar LLMs para reducir el esfuerzo manual en tareas de codificación y documentación, funciones alineadas con la automatización de extracción de requerimientos.

En el estudio de [10]se analizan las técnicas de minería de issue trackers en plataformas colaborativas, destacando cómo el análisis de texto libre en incidencias puede extraer patrones recurrentes, detectar temas críticos y sugerir acciones correctivas. Este enfoque tiene una similitud con el diseño del agente inteligente, donde se pretende recuperar información técnica desde documentos DERCAS en issues de GitLab.

Por otra parte en el caso de [29] se basa en proponer un modelo en el que detectan deudas técnicas de issues dentro de este caso se demuestra cómo el procesamiento semántico de texto en plataformas colaborativas puede aportar valor adicional al ciclo de desarrollo, automatizando actividades que usualmente requerirían revisión humana.

En el siguiente caso de estudio sobre implementación de agentes inteligentes en ingeniería de software se evidencia que están transformando el desarrollo mediante la automatización de tareas, de procesos y mejora en la toma de decisiones. facilitando la automatización y mejorando la eficiencia en diversas áreas.

Un caso relevante es el uso de agentes en la detección de errores y pruebas automatizadas. Estos agentes pueden simular el comportamiento del usuario y ejecutar pruebas de manera continua, lo que permite detectar fallos en etapas tempranas del desarrollo. Además, en la gestión de proyectos, los agentes inteligentes pueden optimizar la asignación de recursos y el seguimiento del progreso, mejorando la colaboración entre equipos.

En la investigación de [30] presenta un enfoque novedoso que utiliza LLM para mejorar el modelo de software, utilizando un aprendizaje rápido, este método admite varias actividades de modelado sin datos de entrenamiento extensos. Inicialmente se centran en formalismos estáticos y conductuales como los diagramas UML, pero su objetivo es extenderlo a otros paradigmas e integrarlo en la línea de Ingeniería dirigida por modelos. Además de evaluar la productividad, la calidad del modelo y la precisión al recibir sugerencias en tiempo real y sensibles al contexto durante las tareas de modelado.

CAPÍTULO 2

2. METODOLOGÍA.

2.1 Definición de la metodología aplicada

El presente trabajo se desarrolló bajo un enfoque metodológico no experimental, de tipo transversal, lo que permitió analizar la situación actual del proceso de levantamiento de requerimientos durante la etapa de creación de requerimientos en una empresa de telecomunicaciones. El propósito de esta fase fue obtener una visión más clara sobre los principales problemas que enfrentaban los usuarios al momento de buscar o referenciar requerimientos previos. Entre los aspectos más críticos se identificaron: ambigüedad en la redacción de los requerimientos, desalineación entre las áreas de negocio involucradas, y pérdida de información histórica en proyectos ya puestos en producción.

A partir de esta evaluación, se planteó el diseño de una solución tecnológica basada en un sistema de búsqueda inteligente, con el objetivo de facilitar a los usuarios el acceso a información previa de forma contextualizada, clara y útil, que sirviera de apoyo en la elaboración de nuevos requerimientos.

2.1.1 Justificación

El método que se eligió en este estudio se fundamenta en la definición del problema planteado y en la forma en que la información está estructurada y disponible dentro de la organización. El trabajo se centra en el diseño de un agente inteligente capaz de facilitar información de requerimientos técnicos realizados en la empresa de telecomunicaciones, lo cual exige un enfoque que combine procesamiento automático, lenguaje natural y recuperación de información.

Esta estrategia metodológica permitió abordar adecuadamente el problema, ya que no solo automatiza el procesamiento de la información, sino que además mejora la forma en que los usuarios acceden al conocimiento existente dentro de los sistemas de gestión de requerimientos.

Adicional no solo se alineó con los objetivos generales y específicos del estudio, sino que además traza una ruta clara para su futura implementación, validación y aplicación en entornos reales de gestión de requerimientos.

La arquitectura propuesta se basa en evidencia técnica documentada, que valida el uso de modelos generativos conectados a motores de recuperación como alternativa eficaz frente a los enfoques tradicionales de preguntas y respuestas [31]. Para realizar la búsqueda semántica, se seleccionó la base de datos vectorial **Qdrant**, por su capacidad para manejar vectores de alta dimensión y su compatibilidad con metadatos. Estas características la convierten en una herramienta ideal para sistemas RAG escalables [32]. GitLab permite configurar Webhooks en proyectos para enviar automáticamente solicitudes HTTP a endpoints externos cuando ocurren eventos relevantes como creación o actualización de issues. Esta capacidad fue aprovechada en el diseño del agente para activar automáticamente los flujos en n8n ante cualquier modificación de requerimientos [33]. Finalmente, [5] subraya el potencial transformador de la Inteligencia Artificial (IA) para optimizar las tareas de gestión de proyectos, lo que respalda la viabilidad de implementar un agente inteligente en la empresa de telecomunicaciones.

Justificación de la alternativa seleccionada.- Esta solución demostró un alto nivel de cumplimiento en los criterios clave del proyecto, destacándose en aspectos como automatización, escalabilidad y reducción del tiempo de búsqueda. En contraste, la alternativa basada únicamente en indexación tradicional obtuvo la puntuación más baja, reflejando su incapacidad para enfrentar los retos semánticos del problema identificado. La segunda alternativa, que proponía el uso de modelos LLM sin automatización, mostró un desempeño intermedio, con buen entendimiento del lenguaje natural, pero limitada por la intervención manual requerida por parte del usuario.

La alternativa seleccionada, al integrar un modelo LLM dentro de un flujo automatizado (N8N), ofreció una solución innovadora, adaptable y con alto impacto en términos de eficiencia operativa. Asimismo, permite escalar su implementación a otras áreas del negocio, aprovechar repositorios existentes como GitLab, y facilitar la consulta contextualizada de requerimientos previos sin necesidad de experiencia técnica por parte del usuario final.

2.2 Recopilación de datos

2.2.1 Levantamiento información mediante encuestas

Recopilación y análisis de la información.- Para recopilar la información necesaria, se diseñó un cuestionario estructurado con preguntas cerradas, orientadas a identificar los principales puntos de dolor relacionados con el acceso a información histórica, el funcionamiento actual del repositorio y la percepción sobre la viabilidad de implementar una solución automatizada. La encuesta estuvo dirigida a actores clave dentro del proceso de levantamiento de requerimientos, incluyendo:

- 12 jefaturas departamentales responsables de la generación de requerimientos.
- 4 analistas de negocio.
- 8 líderes de proyecto.

Lo que representó un total de 24 participantes, correspondiente al 100 % de la muestra definida.

Las encuestas fueron aplicadas mediante la herramienta Google Forms, lo cual permitió recopilar los datos de forma ágil, estructurada y estandarizada. Cada formulario estuvo compuesto por 10 preguntas cerradas, lo que facilitó el posterior análisis cuantitativo de las respuestas. Las preguntas se centraron en evaluar la calidad del proceso actual de búsqueda de requerimientos, así como el tiempo promedio que los usuarios tardaban en encontrar información relevante dentro del repositorio institucional.

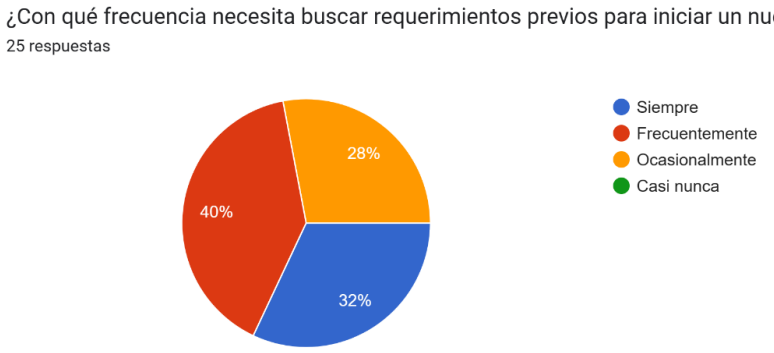
Una vez completada la fase de recolección de datos, se procedió a su análisis utilizando herramientas estadísticas básicas. A través del cálculo de frecuencias y porcentajes, fue posible agrupar las respuestas por bloque temático y detectar patrones comunes que evidenciaron los principales puntos críticos del proceso. Los resultados fueron representados mediante gráficos que permitieron visualizar de manera clara la percepción de los distintos perfiles encuestados, tanto técnicos como no técnicos. Esta diversidad de perspectivas facilitó una validación más completa del diagnóstico y una comprensión integral del problema.

Entre los principales hallazgos, se identificó que un número considerable de usuarios experimentaba dificultades frecuentes para localizar requerimientos previos, lo cual repercutía en la eficiencia del proceso. Asimismo, se evidenció la necesidad recurrente de realizar múltiples iteraciones entre los usuarios y los responsables del proceso antes de consolidar el requerimiento en el formato DERCAS, como consecuencia de la ausencia de mecanismos que permitieran consultar de manera rápida y estructurada los requerimientos documentados en proyectos puestos en producción. Esta limitación obligaba a los actores a recurrir a consultas informales. Estos resultados permitieron confirmar que existía una oportunidad clara de mejora mediante el uso de herramientas tecnológicas que optimicen la búsqueda y recuperación de información.

Adicionalmente, se llevó a cabo una evaluación comparativa entre el tiempo promedio de búsqueda reportado por los usuarios en las encuestas y el tiempo estimado de consulta de un agente inteligente basado en modelos de lenguaje natural (LLM). Esta comparación se fundamentó en referencias documentadas de soluciones similares ya implementadas, donde se evidenciaron mejoras en la recuperación de información. Para el análisis se consideraron dos indicadores clave:

- El tiempo promedio que un usuario tardaba actualmente en localizar un requerimiento funcional específico.
- El tiempo estimado de respuesta de una API conectada al agente inteligente, capaz de entregar resultados relevantes de forma automatizada y contextualizada.

Para sustentar esta estimación, se revisaron estudios recientes como el de [34], en el cual se analizaron estrategias de optimización aplicadas durante la ejecución de modelos de lenguaje, tales como el escalado dinámico en tiempo de consulta (latency-aware test-time scaling), que permitieron reducir significativamente los tiempos de respuesta sin comprometer la precisión de los resultados. Esta evidencia respaldó la viabilidad técnica de implementar un agente inteligente enfocado en la búsqueda eficiente de requerimientos.



.

Fig. 1. Frecuencia con la que los encuestados necesitan buscar requerimientos.

Análisis e interpretación: La mayoría de los encuestados (72 %) recurrió con frecuencia a requerimientos previos, lo que evidenció una alta dependencia de información histórica.

Fig. 2. Tiempo estimado para localizar requerimientos

Análisis e interpretación: El 40 % de los encuestados indicó que tardaba más de 15 minutos en encontrar un requerimiento, lo que evidenció dificultades en el acceso y búsqueda.

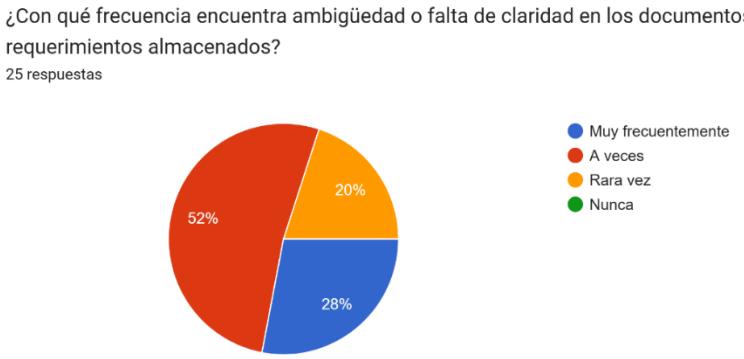


Fig. 3. Frecuencia con que se detecta ambigüedad en los requerimientos

Análisis e interpretación: El 80 % de los encuestados reportó encontrar falta de claridad en los documentos.

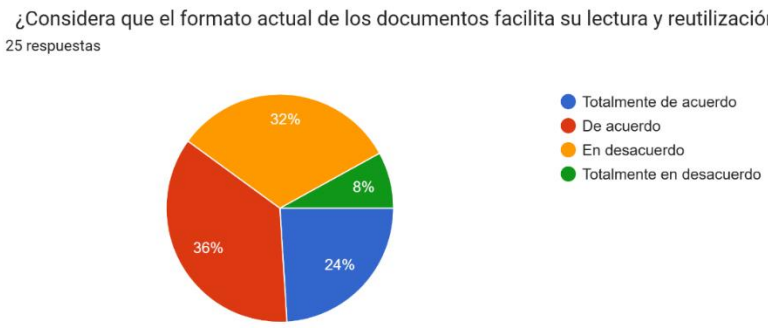


Fig. 4. Opinión sobre el formato actual de los documentos de requerimientos.

Análisis e interpretación: El 60 % expresó un grado de conformidad con el formato actual, un 40 % manifestó desacuerdo, lo que indicó oportunidades de mejora en la estructura documental.

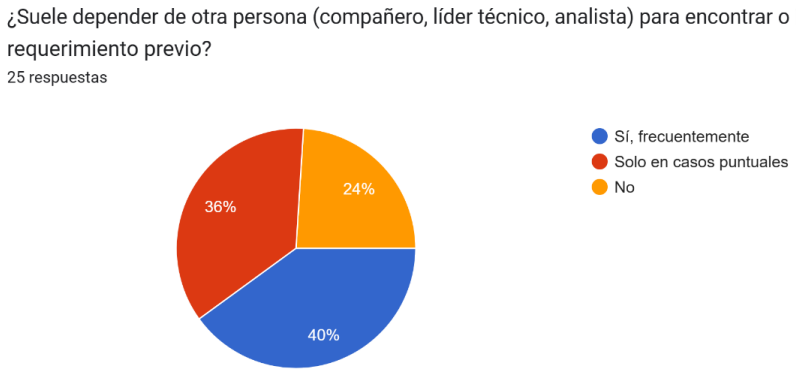


Fig. 5. Dependencia de información de otra persona.

Análisis e interpretación: El 76 % reconoció depender de otros, para interpretar requerimientos, lo que evidenció una necesidad de mejorar la autonomía en el acceso de los documentos.

Fig. 6. Opinión de los encuestados sobre la organización de los requerimientos.

Análisis e interpretación: Solo el 12 % percibió una organización lógica en el repositorio actual, mientras que el 88 % expresó desacuerdo, lo que reflejó una mejora en la gestión de requerimientos.

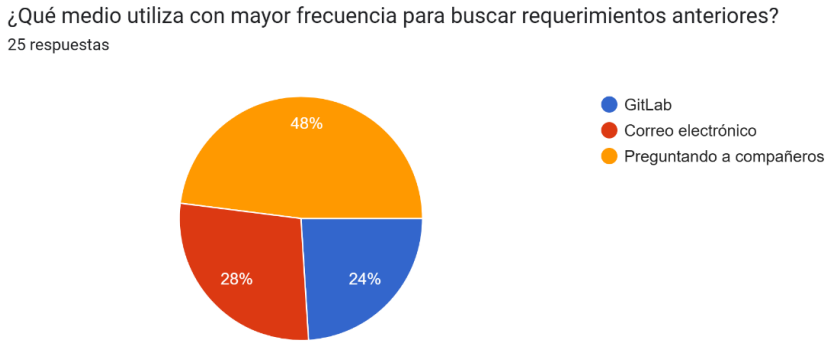


Fig. 7. Medio más utilizado por los encuestados para buscar requerimientos.

Análisis e interpretación: El 48 % de los encuestados buscó requerimientos previos consultando a compañeros.

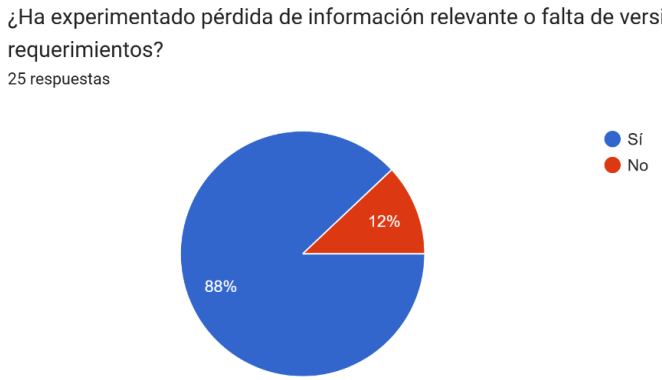


Fig. 8. Incidencia de pérdida de información de los requerimientos.

Análisis e interpretación: El 88 % indicó haber experimentado pérdida de información.

Fig. 9. Percepción sobre un sistema de búsqueda inteligente basado en lenguaje natural para recuperar requerimientos.

Análisis e interpretación: El 88 % consideró necesario implementar un sistema de búsqueda inteligente, lo que reflejó un alto interés por soluciones que optimicen el acceso a requerimientos

.

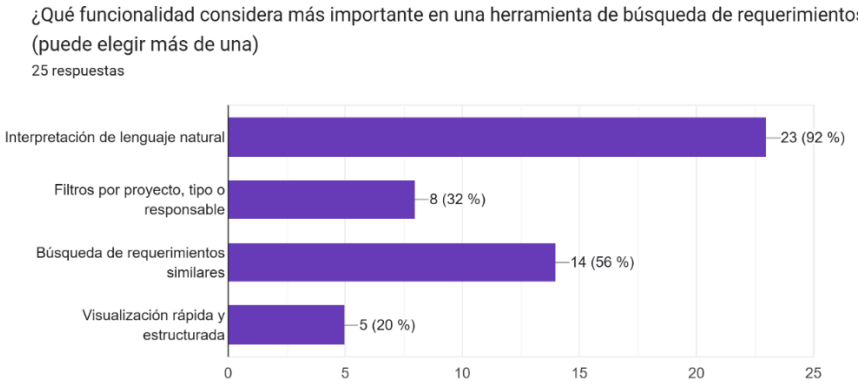


Fig. 10. Puntuación de funcionalidades más importantes en una herramienta de búsqueda de requerimientos.

Análisis e interpretación: La funcionalidad más solicitada fue la interpretación de lenguaje natural (92 %), seguida de la búsqueda de requerimientos similares (56 %), lo que confirmó la necesidad de herramientas intuitivas y orientadas al contexto de la información de requerimientos.

2.2.2 Proceso actual AS-IS

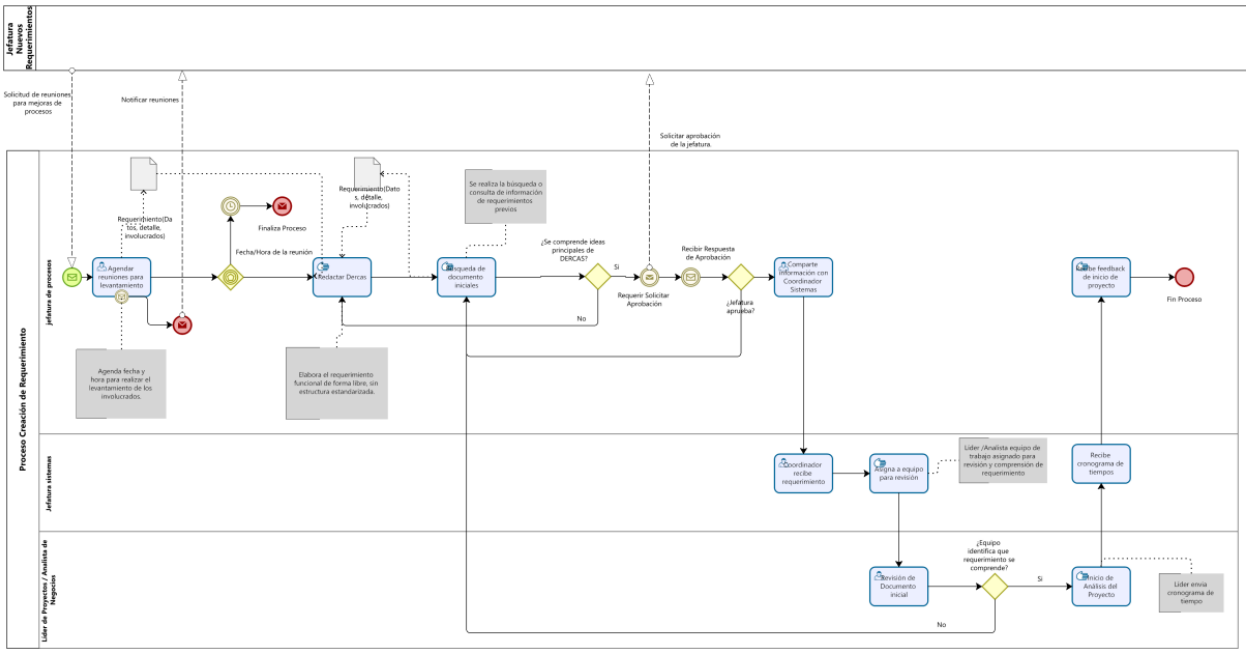


Fig. 11. Flujo actual (AS-IS) de creación de requerimientos.

El proceso de creación de requerimientos, se iniciaba cuando la jefatura de un departamento identificaba la necesidad de una mejora o nuevo requerimiento. Como primer paso, la jefatura se reunía con el departamento de procesos con el objetivo de coordinar la agenda inicial para entender el proceso y aplicar la mejora. Una vez que se realizaba la reunión con los involucrados, se redactaban los primeros datos del requerimiento funcional, en el documento que actualmente se maneja llamado DERCAS. Como parte de esta etapa se realizaba un paso fundamental que es la revisión de procesos anteriores o de requerimientos ya en producción. Esta actividad tenía como propósito aprovechar la información inicial y evitar la duplicación de esfuerzos, así como considerar aspectos técnicos ya identificados en requerimientos anteriores.

Una vez documentadas las ideas principales del requerimiento, y siempre que estas fueran claras y completas, el documento era enviado para aprobación a la jefatura responsable de la mejora. En caso de que el contenido no fuera suficientemente comprensible, se procedía a revisar nuevamente documentación relacionada o a retomar sesiones de revisión con los involucrados en el proceso.

Cuando la jefatura aprobaba el requerimiento, este era compartido con el Coordinador de Sistemas. El mismo que realizaba una revisión preliminar y asignaba el nuevo requerimiento al equipo responsable. Dicha asignación contemplaba la participación de un líder de proyecto, encargado de validar la información técnica, y un analista, quien revisaba la estructura y redacción del documento de requerimiento.

Tras el análisis del documento, si el equipo técnico comprendía correctamente el alcance de lo solicitado, se daba inicio a la etapa de análisis del proyecto. En este punto, el líder de proyecto elaboraba un cronograma estimado de tiempos, el cual era validado por el Coordinador de Sistemas y luego remitido al usuario del área de procesos. Por el contrario, si durante la revisión se detectaban inconsistencias o ambigüedades, el requerimiento era devuelto al sponsor principal para su corrección y reformulación, con el fin de que fuese reenviado en una versión clara.

Dentro de este flujo se evidenció limitaciones relevantes, tales como la falta de herramientas para sistematizar la revisión de requerimientos previos, la ausencia de un repositorio centralizado de requerimientos, y la dependencia de reiteradas reuniones para avanzar en el entendimiento del requerimiento, con esto se propone un flujo TO-BE en el cual se centra en la mejora de estas falencias.

2.3Propuesta del flujo automatizado

2.3.1 Alternativas de solución

Descripción de las alternativas propuestas.- Para determinar cuál de las propuestas representaba la solución más adecuada al problema, se realizó un análisis comparativo mediante la técnica Matriz de Decisión Ponderada, la que permitió evaluar las opciones según criterios, entre tres alternativas técnicas, las cuales se orientaron en la mejora del acceso a los requerimientos funcionales. Esta metodología permitió comparar de forma estructurada las alternativas consideradas, asignando un peso a cada criterio y evaluando el grado de cumplimiento por parte de cada propuesta.

Las alternativas fueron las siguientes:

- **Alternativa 1:** Sistema de indexación sin inteligencia contextual.

Esta opción consistió en implementar un motor de búsqueda que se base en mecanismos de indexación y filtrado por palabras clave. Permitía organizar los requerimientos de manera estructurada por categorías o etiquetas, mejorando la navegación entre el repositorio. Sin embargo, no incorporaba interpretación semántica ni procesamiento del lenguaje natural, lo que limitaba su capacidad para responder consultas o adaptarse al contexto del usuario en la búsqueda.

- **Alternativa 2:** Uso de modelo LLM con consulta directa (sin automatización).

Esta alternativa propuso la incorporación de un modelo de lenguaje natural (LLM) para interpretar preguntas y ofrecer respuestas a partir de la información que se cargue. Esto mejoraba en la comprensión del contenido y permitía formular preguntas en lenguaje natural, pero se dependía completamente del usuario y la manualidad en la carga de archivos, iniciar la consulta y gestionar la interacción.

- **Alternativa 3:** Agente de búsqueda inteligente con automatización mediante flujo N8N.

Esta opción integró el uso de un modelo LLM dentro de un flujo automatizado diseñado en N8N, conectado directamente con los repositorios GitLab. El agente se activaba al detectar nuevos archivos o actualizaciones, extraía la información relevante y permitía realizar consultas en lenguaje natural con respuestas estructuradas. Esta alternativa combinó automatización, interpretación de lenguaje natural y retroalimentación estructurada, adaptándose además a futuras integraciones.

2.3.2 Selección de la Propuesta

Evaluación comparativa mediante Matriz de Decisión Ponderada.- En el proceso de evaluación comparativa, se utilizó una Matriz de Decisión Ponderada para valorar las alternativas propuestas en función de criterios técnicos y operativos. Cada alternativa fue calificada en una escala de 1 a 5, donde 1 representa un bajo cumplimiento y 5 un cumplimiento óptimo respecto a cada criterio. Esta calificación refleja el nivel de desempeño de cada alternativa.
Cada criterio fue ponderado según su relevancia estratégica y técnica, con pesos definidos por los arquitectos del departamento para asegurar un equilibrio entre factibilidad y valor práctico. La suma total fue 1, con la siguiente distribución:

Tabla IV. Criterios de evaluación y peso de alternativas.

Criterio	Peso asignado
Integración con plataformas existentes	0.15
Facilidad de implementación	0.10
Nivel de automatización alcanzable	0.20
Escalabilidad	0.15
Reducción del tiempo de búsqueda	0.25
Costos de desarrollo y mantenimiento	0.15
Total	1.00

Para obtener el puntaje total ponderado de cada alternativa, se multiplicó la calificación por el peso correspondiente en cada criterio, y luego se sumaron todos los resultados. Esta metodología permitió comparar las alternativas de forma cuantitativa y justificada, combinando tanto el cumplimiento técnico como la relevancia estratégica de cada dimensión evaluada.

Tabla V. Evaluación ponderada de alternativas tecnológicas para búsqueda de requerimientos.

Criterio	Peso	Sistema de indexación sin inteligencia contextual	Uso de modelo LLM con consulta directa (sin automatización)	Agente de búsqueda inteligente con automatización mediante flujo N8N
Integración con plataformas	0.15	2 (0.30)	3 (0.45)	5 (0.75)
Facilidad de implementación	0.10	4 (0.40)	3 (0.30)	4 (0.40)
Nivel de automatización	0.20	1 (0.20)	2 (0.40)	5 (1.00)
Escalabilidad	0.15	2 (0.30)	4 (0.60)	5 (0.75)
Reducción del tiempo de búsqueda	0.25	2 (0.50)	4 (1.00)	5 (1.25)
Costos de desarrollo/mantenimiento	0.15	5 (0.75)	3 (0.45)	4 (0.60)

Puntaje total ponderado	1.00	2.45	3.20	4.75

Revisión de la mejor alternativa. - Los criterios más relevantes fueron la escalabilidad, la automatización y la reducción del tiempo de búsqueda, por su alineación con los problemas detectados inicialmente. Aunque también se consideraron el costo y el tiempo de desarrollo, su ponderación fue menor. La alternativa de indexación sin inteligencia contextual obtuvo el puntaje más bajo por su enfoque tradicional. La opción con LLM sin automatización logró una puntuación intermedia, limitada por su dependencia manual. En cambio, la propuesta con agente inteligente automatizado mediante N8N alcanzó la mayor valoración (4,75), destacando por su escalabilidad, integración y capacidad de automatización. Se concluyó que esta era la opción más adecuada técnica y estratégicamente, por lo que fue seleccionada como base del diseño de solución.

2.3.3 Diseño del flujo

Flujo envío de información Gitlab- Qdrant

Fig. 12. Flujo envío información de Gitlab

El diseño definió un flujo automatizado para el procesamiento de requerimientos desde su creación o edición en GitLab. Su objetivo fue integrar documentos en una base de datos vectorial, permitiendo búsquedas semánticas mediante un modelo de lenguaje LLM. Implementado en N8N, este flujo actuó como orquestador y ejecutó automáticamente operaciones como validación, descarga, extracción, vectorización y almacenamiento tras cada evento registrado en GitLab. A continuación, se describen sus componentes, herramientas y funciones principales.

Evento GitLab.- Al crear o editar un *issue* en GitLab, se activaba automáticamente un webhook previamente configurado, que enviaba los datos del evento al sistema automatizado. Esta configuración se realizaba en *Projects → Settings → Webhook*, donde se registraba la URL del flujo en N8N y, de forma crucial, se activaba el evento *issues*, indispensable para que el flujo se ejecutara correctamente.

Para asegurar una automatización efectiva y facilitar la búsqueda por parte del agente, se definieron estándares que cada *issue* debía cumplir. Estos incluían:

- **Título:** Debe ser corto y bien especifico.
- **Descripción:** Un resumen claro del motivo de creación del issue.
- **Documento adjunto:** Documento adjunto con un nombre especifico “DERCAS.pdf”.
- **Label:** Para este diseño se le asociaría “Requerimiento”.

Fig. 13. Creación webhooks

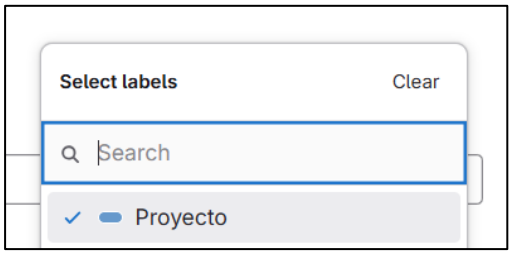


Fig. 14. Creación Labels.

- **Webhook en N8N.-**El flujo en N8N comienza al recibir ese webhook vía POST. El nodo inicial captura los datos enviados por GitLab, incluyendo el project_id y el id del issue. Este evento es en el nodo inicial del flujo N8N.
- **Obtención detalles.-**Con los identificadores del proyecto y del id issue, el flujo realiza una solicitud HTTP a la API REST de GitLab para obtener información completa del requerimiento: título, descripción, etiquetas y enlaces a archivos adjuntos. Todo esto realizado mediante una configuración realizada en este nodo donde tiene la url del api de Gitlab.
- **Validar nombre del archivo.-**Se analiza el contenido de la descripción del issue para extraer la URL del archivo PDF. Se valida que el archivo tenga exactamente el nombre del archivo a extraer. Si no coincide, el flujo se detiene. Esta validación se realiza dentro del mismo nodo que extrae el archivo.
- **Descargar documento.-**Si el archivo es válido, se realiza una descarga del documento desde GitLab mediante una solicitud HTTP. El archivo PDF, con esto se enviará al siguiente nodo para que continue el flujo
- **Preparar metadatos.-**Se genera un objeto JSON que contiene los metadatos relevantes del requerimiento, incluyendo el título del issue, el nombre del proyecto, la fecha del evento y la URL del archivo. Este JSON acompañará al archivo para su procesamiento.
- **Procesar documento.-**Se realiza una solicitud HTTP POST al endpoint /procesar_pdf, enviando el archivo más los metadatos. El método recibe este contenido y comienza el procesamiento.
- Este método extrae el texto del documento PDF usando la herramienta pdfplumber, luego convierte el texto en vectores semánticos teniendo lista la información para bitacorizar en Qdrant.
    - Debido a que la información obtenida es extensa, esta información se divide en fragmentos de 200 palabras máximo, se detalla ejemplo.

Tabla VI. Fragmento palabras

Fragmento de 200 palabras

{
"id": "frag-001",
"vector": [0.12, 0.98, ...],
"payload": {
"titulo": "Proyecto Pedidos",
"tipo": "Requerimiento",
"fecha": "2024-09-15",
"contenido": "El sistema permite crear pedidos para el personal de campo……."
}
{
….
}

- Para generar los vectores semánticos, se utilizó el modelo all-MiniLM-L6-v2 que es una librería, que transforma texto en vectores de 384 dimensiones. Este modelo fue seleccionado por su eficiencia, precisión y compatibilidad con sistemas de búsqueda semántica como Qdrant.
- Una vez generados estos fragmentos son enviados para su almacenamiento tal como se indica en el siguiente punto.
- **Guardar en Qdrant.-** Finalmente, los vectores generados junto con su contexto se almacenan en la base de datos vectorial Qdrant. Esto permite realizar búsquedas semánticas por similitud en futuras consultas de usuario.

- Antes de poder hacer los registros se creó la colección donde se determina la distancia Coseno que es la más adecuada para Qdrant garantizando una recuperación más precisa y coherente del contenido de información
- A continuación, se detalla un ejemplo grafico de la vectorización y almacenamiento.

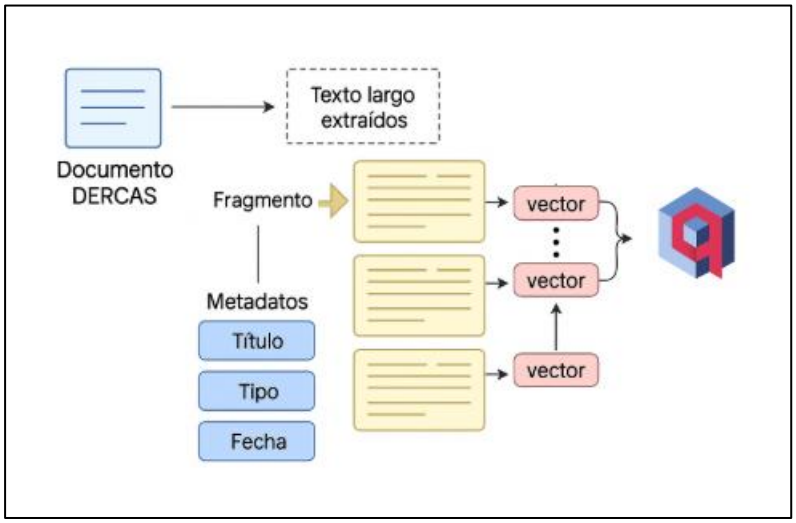


Fig. 15. Flujo vectorización en qdrant.

Para los proyectos actuales se analizó generar la información histórica de estos con un solo documento integrado por proyecto que fue proporcionado por el departamento de procesos que realizó esta tarea por el mismo tema de que no se tenía información a la mano de mayoría de estos proyectos, estos documentos contienen las diferentes funcionalidades que se aplicaron a cada uno de estos. En conclusión, una vez obtuvo estos documentos se agregaron a un issues cumpliendo así con la carga automática al dispararse el webhook que inicia este flujo.
A continuación, se detalla un gráfico del flujo en N8N.

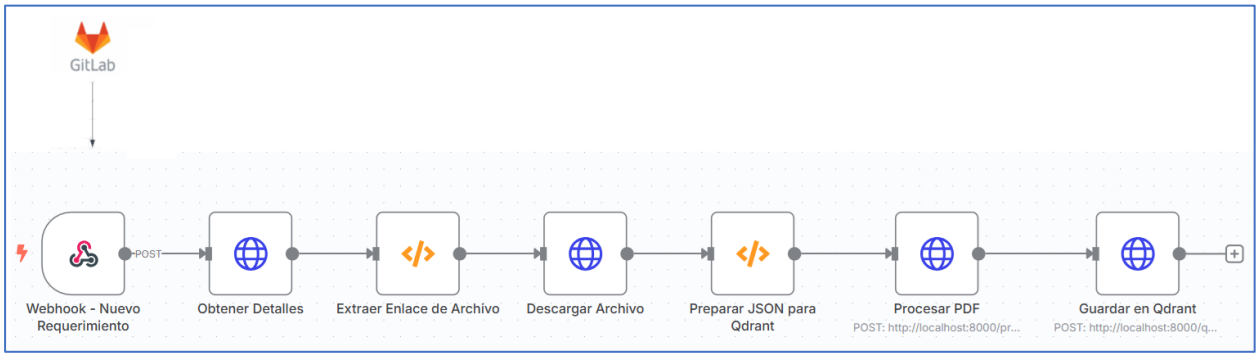


Fig. 16. Flujo N8N automatización.

2.3.4 Herramientas

Tabla VII. Herramientas y Uso.

Herramienta	Uso en el diseño
N8N	Orquestador de flujos de automatización
GitLab API	Fuente de datos (issues y adjuntos como DERCAS)
Qdrant	Base de datos vectorial para almacenamiento y recuperación por similitud
LLama	Generación de respuestas en lenguaje natural a partir del contexto
Pdfplumber	Biblioteca de extracción de texto desde documentos PDF
all-MiniLM-L6-v2	Librería para generar vectores de carga y consulta
Bizagi Modeler	Para modelo de flujo de procesos

2.3.5 Especificaciones.

Técnicas. - Este diseño se basa en una arquitectura modular e integrable, orientada a facilitar el acceso a información de requerimientos funcionales dentro de una empresa de telecomunicaciones. Entre las principales características técnicas del sistema propuesto se incluyen:

- Automatización de flujos con N8N para la recolección estructurada de datos y documentos asociados desde GitLab mediante API REST.
- El flujo n8n puede incorporar nodos de validación y reintento para manejar errores como respuestas vacías, formatos malformados o caídas temporales del servidor LLM o la base vectorial.
- Técnicas de vectorización en Qdrant y modelos de lenguaje con Llama, que permiten representar el contenido de los requerimientos de manera semántica.
- Para mejorar la calidad de la vectorización, los documentos extraídos desde GitLab son divididos en fragmentos de tamaño razonable, asegurando que los vectores representen unidades coherentes de significado.
- Una arquitectura basada en RAG (Retrieval-Augmented Generation), que combina búsqueda vectorial en Qdrant con generación de respuestas en lenguaje natural.
- El diseño utiliza el modelo Llama para la generación de respuestas en lenguaje natural, ejecutado en el entorno del proyecto, sin depender de servicios externos.

Consideraciones Éticas y Legales

- Todos los datos utilizados en esta investigación provienen de entornos controlados, sin afectar operaciones reales ni comprometer información sensible de la empresa.
- No se procesaron datos sensibles ni personales de usuarios, conforme a lo establecido en normativas de protección de datos. La información fue limitada a requerimientos funcionales, sin incluir nombres, identificadores u otra información personal.
- Se utilizaron exclusivamente herramientas open-source (como GitLab, Qdrant, N8N y Llama), en cumplimiento con sus respectivas licencias de uso y términos comunitarios.
- El diseño no expone datos hacia internet ni usa servicios de terceros. Todo el procesamiento debe ser dentro de redes privadas, lo que cumple con políticas de confidencialidad de empresas de telecomunicaciones.
- El modelo de lenguaje no fue ajustado con datos sensibles ni entrenado con información privada. Su uso se limita a generación de lenguaje sobre datos previamente controlados.
- Se considera la necesidad de establecer gobernanza de datos a futuro para el uso de inteligencia artificial dentro de la organización. Aunque este diseño no abarca directamente una política de gobernanza, se reconoce la importancia de definir responsabilidades, límites de uso y criterios éticos en futuras fases de implementación.
- Todas las respuestas generadas pueden ser rastreadas hasta el contexto exacto que las originó, permitiendo así una auditoría y revisión identificando así la transparencia y fiabilidad de la información proporcionada de parte del agente.

2.4 Diseño del agente de búsqueda

2.4.1 Arquitectura

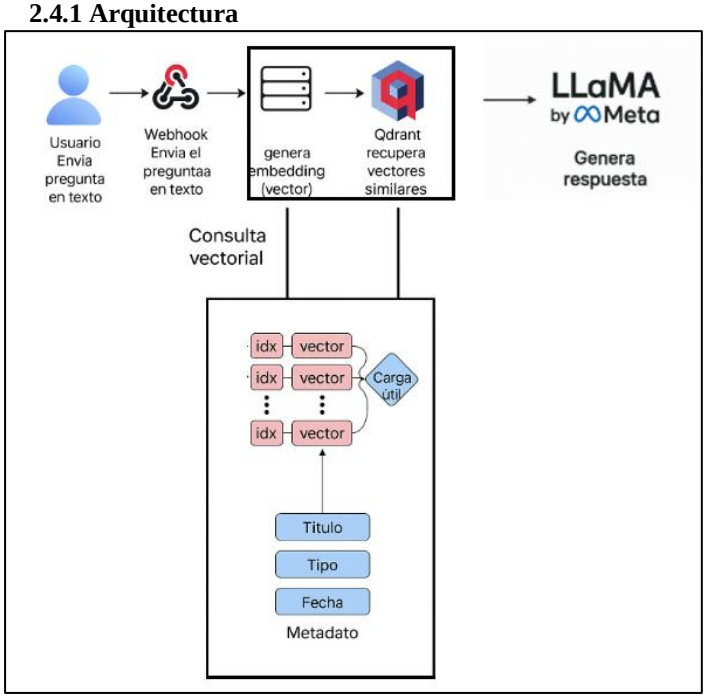


Fig. 17. Diagrama de interacción entre componentes.

- **Consulta** El flujo de consulta inició cuando un usuario realiza una pregunta en lenguaje natural. Esta consulta se envía como una petición HTTP POST al endpoint de N8N que está expuesto en el webhook del flujo de consulta.
- La consulta al sistema puede ser integrada desde cualquier interfaz frontend de la empresa que permita realizar solicitudes HTTP, como aplicaciones web, móviles o bots conversacionales.
- Esto es posible gracias a que el flujo en N8N expone un webhook accesible vía HTTP, el cual acepta datos en formato JSON.
- Al tratarse de una API abierta, cualquier cliente capaz de realizar una petición POST puede enviar la pregunta del usuario y recibir la respuesta generada por el modelo Llama.
- Esta arquitectura facilita la incorporación de la solución en múltiples canales, sin acoplamiento directo con la lógica interna del procesamiento.
- **Webhook - Consulta Usuario. -** El flujo comienza con el nodo Webhook en N8N, el cual recibe una consulta enviada por el usuario. La entrada debe ser una solicitud HTTP POST que contenga el texto de la pregunta en formato JSON bajo el campo 'consulta'.
- **Vectorizar Consulta. –** Convierte la pregunta del usuario a vector usando un endpoint de modelo de embedding SentenceTransformer("all-MiniLM-L6-v2"). A continuación, un ejemplo.

Tabla VIII. Ejemplo de Vector

Vector
"vector": [0.03989684581756592, 0.06593604385852814, -0.010924134403467178, 0.008360879495739937, -0.07364846020936966, 0.03933945298194885, 0.11244940012693405, -0.009591905400156975, 0.06352280080318451]

- **Buscar en Qdrant.-** El contenido de la consulta se envía al endpoint /qdrant/search que interactúa con la base de datos vectorial Qdrant. En este punto, el texto se vectoriza y se ejecuta una búsqueda semántica sobre

los vectores previamente cargados a través del flujo de carga. Teniendo como resultado una lista de texto relevantes que sirven como contexto.

Tabla IX. Vector Consulta y Resultado

El resultado de la búsqueda devuelve información en el siguiente formato.

"result": [
{
"id": 0,
"version": 0,
"score": 0.5519424,
"payload": {
"texto": "El sistema debe permitir la gestión de usuarios a través de roles."
}
}

- **Pasar pregunta. -** Extrae únicamente la pregunta original desde el Webhook y la formatea como nuevo objeto.
- **Unir Pregunta más resultado. –** Este nodo consiste en tener una estructura combinada que contiene tanto la pregunta como los textos relevantes para generar la respuesta. Esta información proviene de los nodoso Pasar pregunta y Buscar en Qdrant, teniendo como resultado un json con la información unificada.
- **Generar Respuesta con Llama. -** La pregunta original del usuario y el contexto recuperado desde Qdrant se envían al endpoint /llama/responder. Este endpoint consulta al modelo Llama, que genera una respuesta textual coherente y relevante utilizando la información semántica recuperada. La respuesta es enviada de vuelta al flujo y puede devolverse al usuario directamente o mostrarse a través de una interfaz.
- A continuación, se detalla una figura del flujo descrito en N8N.
- Limpiar Texto. – Hace una depuración del texto eliminando caracteres y saltos de línea demás dejando un texto disponible para devolver.
- **Responder. -** Devuelve la respuesta generada al cliente original, al tener este tipo de nodo el canal que envié la petición esperará la respuesta del flujo N8N.

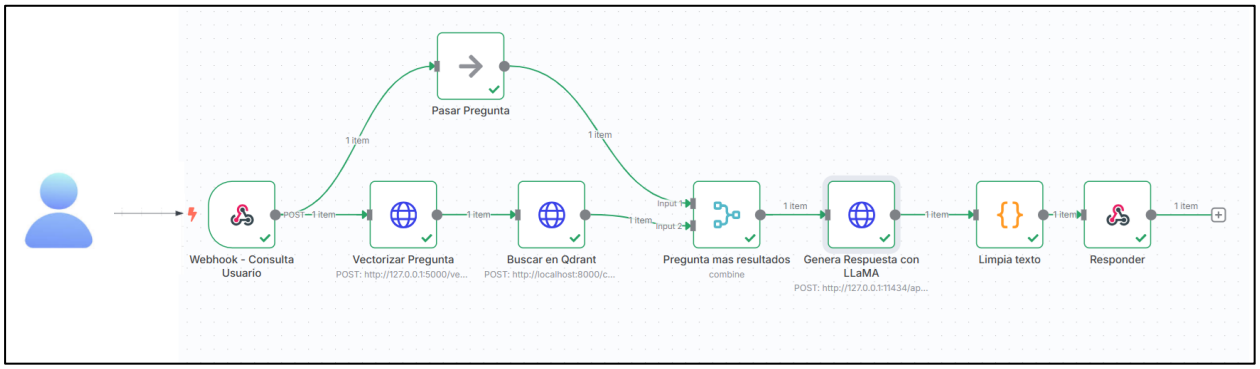


Fig. 18. Diagrama de interacción respuesta Llama

En resumen, se planteó una solución pensada para ayudar a las personas que necesitan acceder rápidamente a información de proyectos anteriores. Esta propuesta busca que el proceso sea más sencillo, claro y ordenado, aprovechando herramientas que permiten encontrar la información sin tener que buscarla manualmente. Además, se pensó de forma flexible para que pueda adaptarse a otras áreas o necesidades similares. En el siguiente capítulo se mostrarán los resultados que se obtuvieron al probar esta idea, y cómo podría mejorar el trabajo que hoy en día hacen los equipos dentro de la organización.

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

3.1 Presentación de la solución

A continuación, se describe la solución propuesta para mitigar el problema identificado, la dificultad de acceso rápido y preciso a los requerimientos funcionales de proyectos en una empresa de telecomunicaciones. La propuesta consiste en el diseño de un agente de búsqueda inteligente, capaz de procesar consultas en lenguaje natural, buscar en una base de datos vectorial, y retornar resultados relevantes y contextualizados. El diseño integra herramientas de automatización, procesamiento de lenguaje natural y recuperación de información semántica, también obtener la información de primera instancia para que mediante un flujo automatizado en N8N envíe esta información a una base en Qdrant de donde se obtendrá la información de las consultas que se realicen.
Esto reduce la dependencia de personal técnicos, mejora la productividad en áreas de análisis y desarrollo, y disminuye la posibilidad de omitir requerimientos críticos.
La solución propuesta se centra en un diseño que demuestre la viabilidad técnica de un sistema automatizado e inteligente para el acceso a requerimientos. La propuesta incluye los siguientes elementos clave:

- Gitlab como fuente estructurada de documentos técnicos y requerimientos oficiales.
- N8N, como flujo automatizado para obtener la información de documentos enviados desde Giltab.
- Qdrant, una base de datos vectorial para el almacenamiento semántico de requerimientos.
- Un modelo de lenguaje LLM, como Llama, para procesar consultas en lenguaje natural.

El sistema inicia con una consulta, por ejemplo: ¿Cuáles son los requisitos para el módulo de facturación digital? Esta entrada es procesada y enviada a través de un flujo automatizado, desarrollado en N8N, que transforma la consulta en un vector numérico representativo.
Este vector se compara contra los vectores previamente almacenados en la base Qdrant, que representa requerimientos obtenidos de los documentos extraídos desde GitLab. Los resultados más relevantes basados en similitud semántica son devueltos al sistema, y una respuesta en lenguaje natural es finalmente presentada al usuario.

3.2 Resultados de validación, pruebas o simulaciones

Se realizaron varias pruebas en un entorno simulado. Estas pruebas del flujo automatizado nos ayudaron a validar el comportamiento del mismo, validar el tiempo que se tarda en dar una respuesta, y si la información que entrega el agente este contextualizada.
Las simulaciones se hicieron a través de Postman, donde se enviaron preguntas al flujo automatizado creado en N8N. Este flujo simuló lo que haría el agente inteligente: recibió la pregunta, buscó la información de los vectores relacionados y generó una respuesta basada en la información disponible.
A continuación, se muestran los principales resultados obtenidos durante estas pruebas:

- Tiempos de respuesta por cada parte del flujo. - Se midió cuánto tiempo toma cada paso desde que el sistema recibe una consulta hasta que entrega la respuesta final. En promedio, todo el proceso duró menos de 6 minutos, lo cual representa una mejora significativa frente al proceso actual, donde las personas pueden tardar más de 1 día buscando información similar o incluso no llegar a encontrarla.

Tabla X. Tiempos de respuesta en Flujo de Automatización N8N

Etapa del flujo	Tiempo promedio (segundos)
Vectorización de la consulta	10–15

Búsqueda semántica en Qdrant	8–12
Procesamiento de resultados	10
Generación de respuesta con LLaMA	240–300
Limpieza del texto y respuesta final	15–20
Total del flujo (promedio)	**360 segundos (6 minutos)**

Estas son algunas de las preguntas enviadas al sistema y las respuestas que generó en cada caso. También se indica el tiempo que tomó procesar cada una:

Tabla XI. Tiempo de respuesta entre consulta

Consulta realizada	Tiempo de respuesta	Respuesta generada
¿Qué requerimientos tiene el proyecto de pedidos?	44 segundos	Módulos de solicitud, aprobación y seguimiento de órdenes.
¿Qué se implementó en trazabilidad?	47 segundos	Geolocalización y registro de eventos por activo.
¿Quién lideró el módulo de instalación?	46 segundos	Responsable: Coordinador de Operaciones.

Estas respuestas fueron útiles y coherentes con el contenido de los documentos almacenados, lo que demuestra que el sistema puede entregar información relevante sin necesidad de intervención humana.
El nuevo flujo automatizado logró reducir la dependencia de otras personas, acortar tiempos de revisión documental y mejorar la calidad de las respuestas.

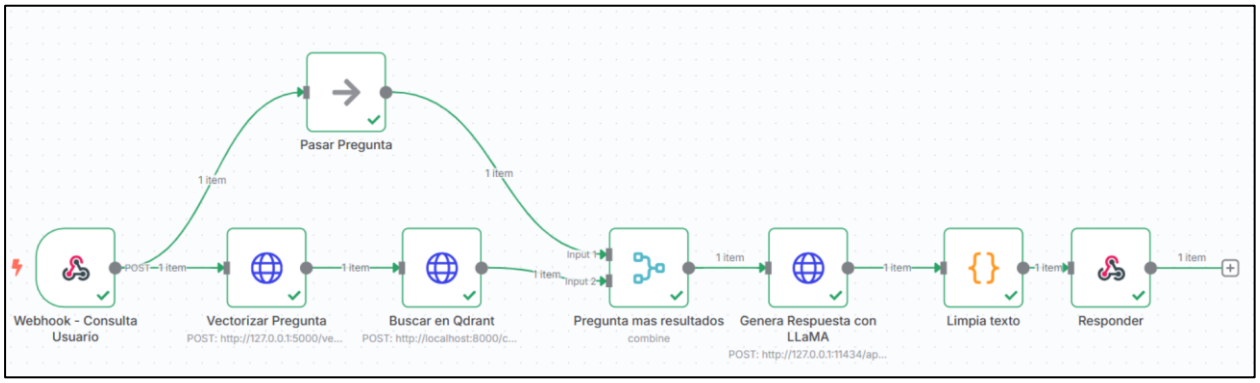


Fig. 19. Flujo automatización ejecutado

Asimismo, para que el sistema pudiera entregar respuestas relevantes, se alimentó previamente la base de datos vectorial con requerimientos reales extraídos de proyectos. Para esta tarea se utilizó un script en Python, que convirtió cada texto en un vector mediante un modelo semántico (SentenceTransformer) y lo insertó en la colección de Qdrant.

```python
from qdrant_client import QdrantClient
from qdrant_client.models import PointStruct
from sentence_transformers import SentenceTransformer

client = QdrantClient(host="localhost", port=8000)
model = SentenceTransformer("all-MiniLM-L6-v2")

collection_name = "requerimientos_tesis"

textos = [
    "El sistema debe permitir la gestión de usuarios a través de roles.",
    "Se requiere implementar un módulo de facturación electrónica.",
    "El chatbot debe responder consultas sobre requerimientos funcionales.",
    "Los reportes deben exportarse en formato PDF y Excel.",
    "Debe haber integración con GitLab para visualizar cambios en requerimientos.",
]

points = []
for idx, texto in enumerate(textos):
    vector = model.encode(texto).tolist()
    points.append(PointStruct(id=idx, vector=vector, payload={"texto": texto}))

client.upsert(collection_name=collection_name, points=points)

print(f"✅ {len(points)} requerimientos insertados en Qdrant.")
```

Fig. 20. Inserción de requerimientos vectorizados en la colección Qdrant

Este proceso permitió tener una base de conocimiento previamente vectorizada, lo cual fue fundamental para que el agente pudiera realizar búsquedas semánticas eficaces y responder consultas con base en similitud contextual.
Como parte del flujo automatizado, se configuró un webhook de tipo POST en la plataforma N8N. Este webhook

actúa como punto de entrada para recibir consultas por parte de los usuarios, ya sea desde herramientas como Postman o desde una interfaz. Cada vez que una pregunta es enviada, el sistema la redirecciona automáticamente hacia los siguientes componentes del flujo para su procesamiento semántico.

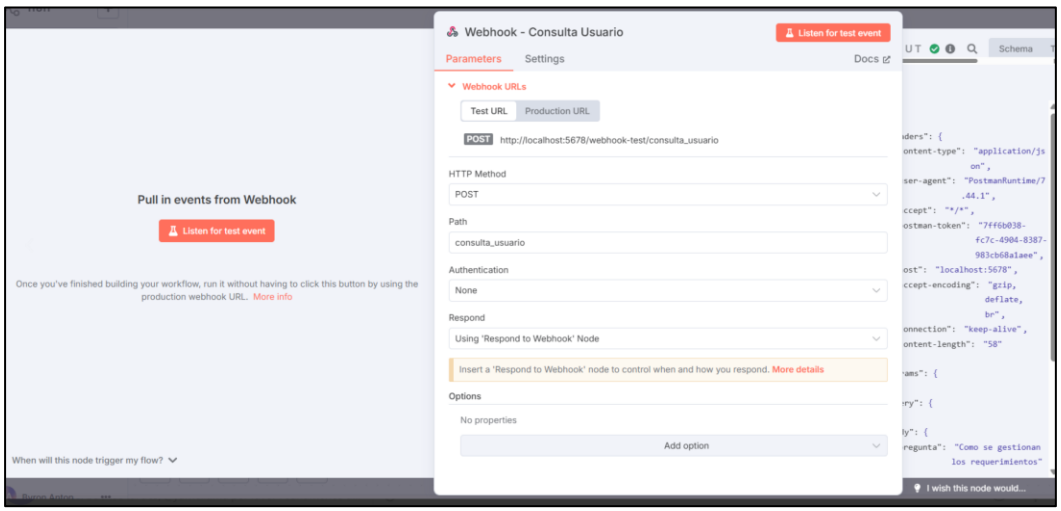


Fig. 21. Configuración del Webhook para recepción de consultas de usuario en N8N

Una vez que el sistema recibe la consulta del usuario, el siguiente nodo consiste en transformar esa pregunta en una representación que pueda ser entendida por el modelo. Para esto, se utiliza un componente encargado de convertir el texto de la pregunta en una secuencia de números. Esta transformación permite que la información pueda ser comparada con otras ya almacenadas, buscando así similitudes que ayuden a encontrar respuestas más acertadas.
El sistema traduce la pregunta a un lenguaje interno, basado en números llamado conjunto de vectores, que le facilita identificar patrones comunes entre preguntas nuevas y antiguos requerimientos.

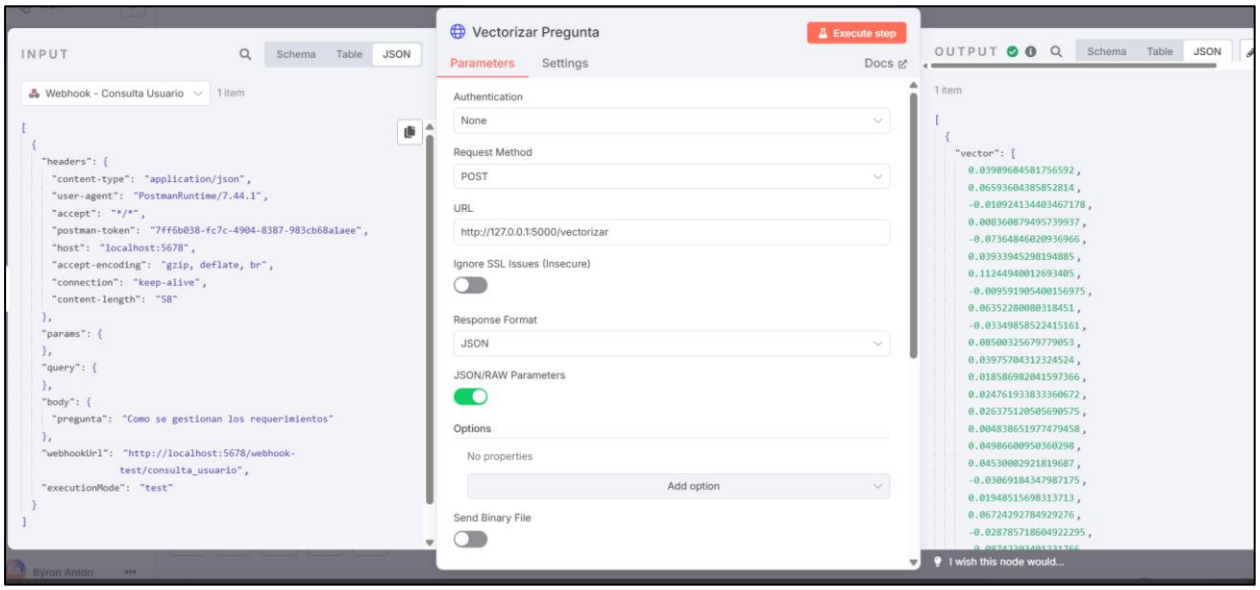


Fig. 22. Transformación de la pregunta del usuario en una representación numérica para su análisis

Después de transformar la pregunta del usuario en un conjunto de valores numéricos, el sistema compara esa información con los datos previamente almacenados en su base de conocimiento. Este proceso se conoce como búsqueda semántica, y lo que hace es encontrar coincidencias en función del significado, no necesariamente de las palabras exactas. En este caso, el sistema consultó la colección de requerimientos en Qdrant y encontró varios fragmentos relevantes que tenían relación con la consulta enviada.

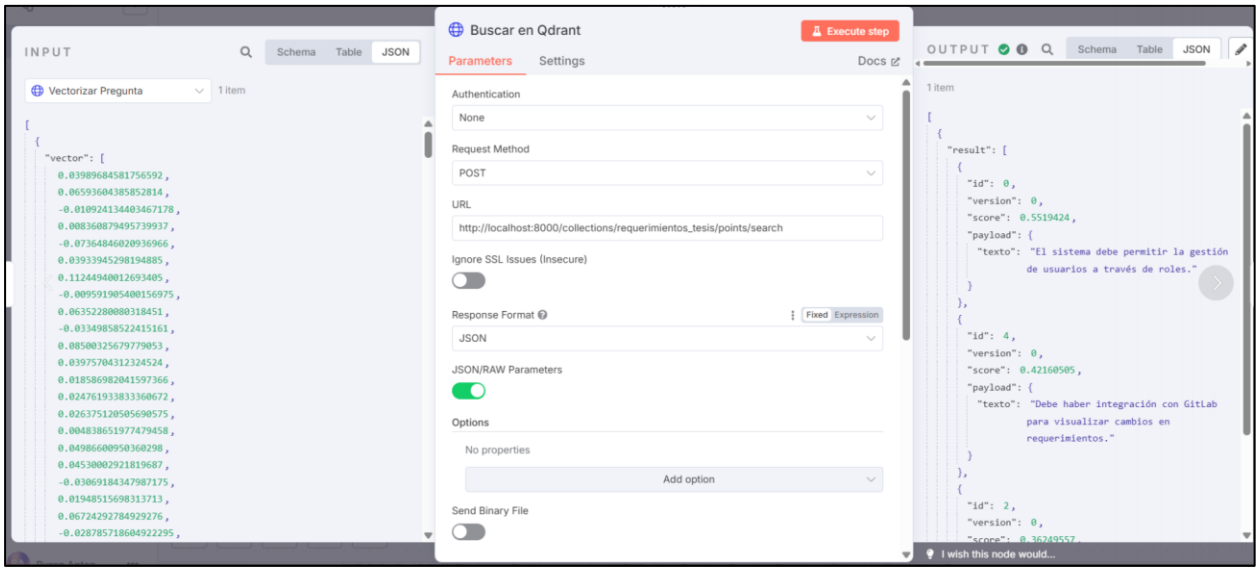


Fig. 23. Búsqueda semántica en la colección en Qdrant

Una vez identificados los resultados más cercanos en la base de conocimiento, el sistema necesita reorganizar los datos para que puedan ser utilizados en la siguiente etapa. Este nodo toma la información recibida y la prepara para continuar con el flujo.
Este nodo, aunque no realiza ninguna transformación, es fundamental para mantener la secuencia lógica del flujo, asegurando que tanto la pregunta original como los textos encontrados estén disponibles para que el sistema pueda generar una respuesta completa y coherente.

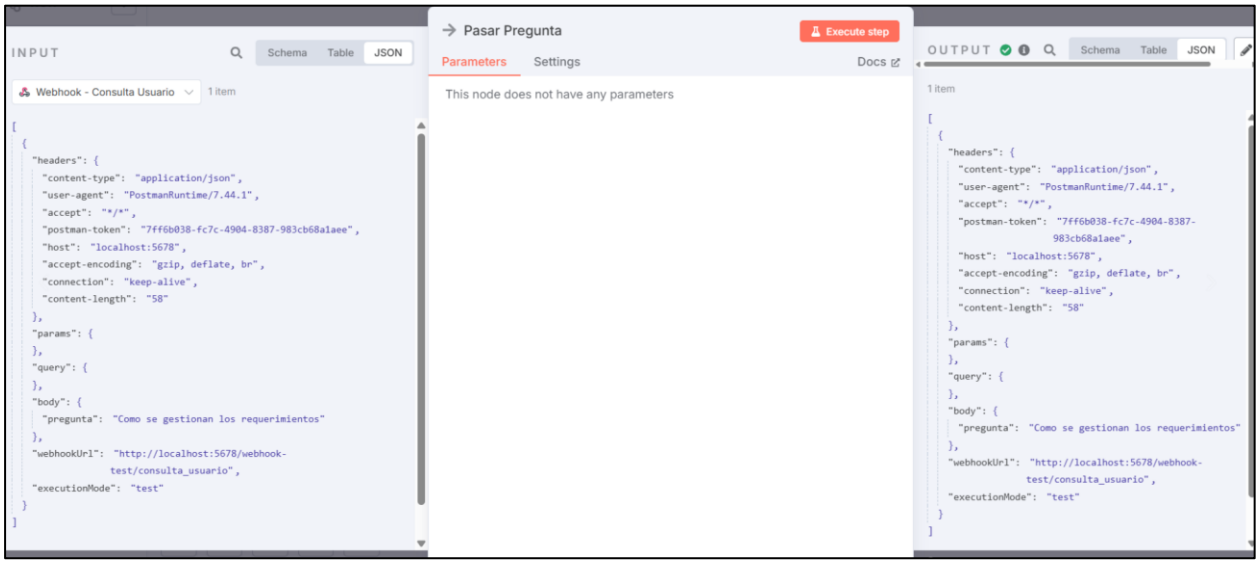


Fig. 24. Nodo que transfiere la pregunta y los resultados hacia el generador de respuesta

Después de recuperar los textos más relacionados con la pregunta, el sistema los une con la consulta original. Esta combinación permite entregar al modelo toda la información relevante, asegurando que la respuesta final sea clara, coherente.
En esta etapa, el sistema construye una respuesta utilizando un modelo de lenguaje. Para que esta generación sea precisa, se definió un mensaje inicial conocido como **prompt** que le indica al modelo que debe responder únicamente con base en los textos entregados como contexto, sin inventar ni salirse del tema. Esta técnica ayuda a asegurar que las respuestas sean claras, útiles y directamente relacionadas con lo que el usuario consultó.

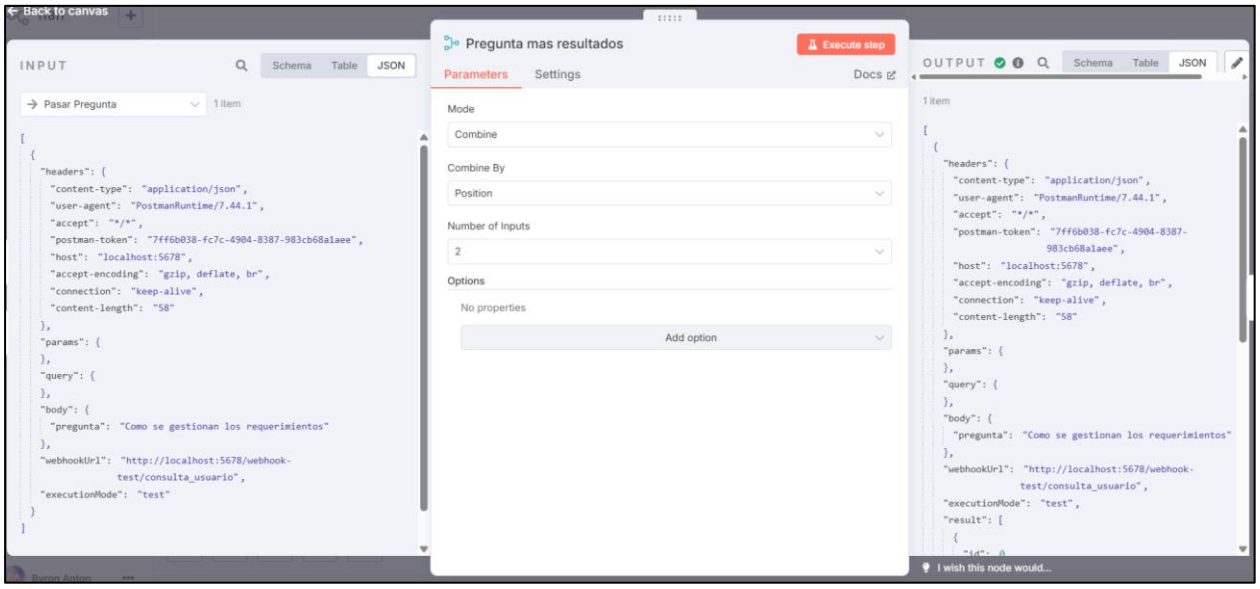


Fig. 25. Combinación de la pregunta original con los resultados encontrados

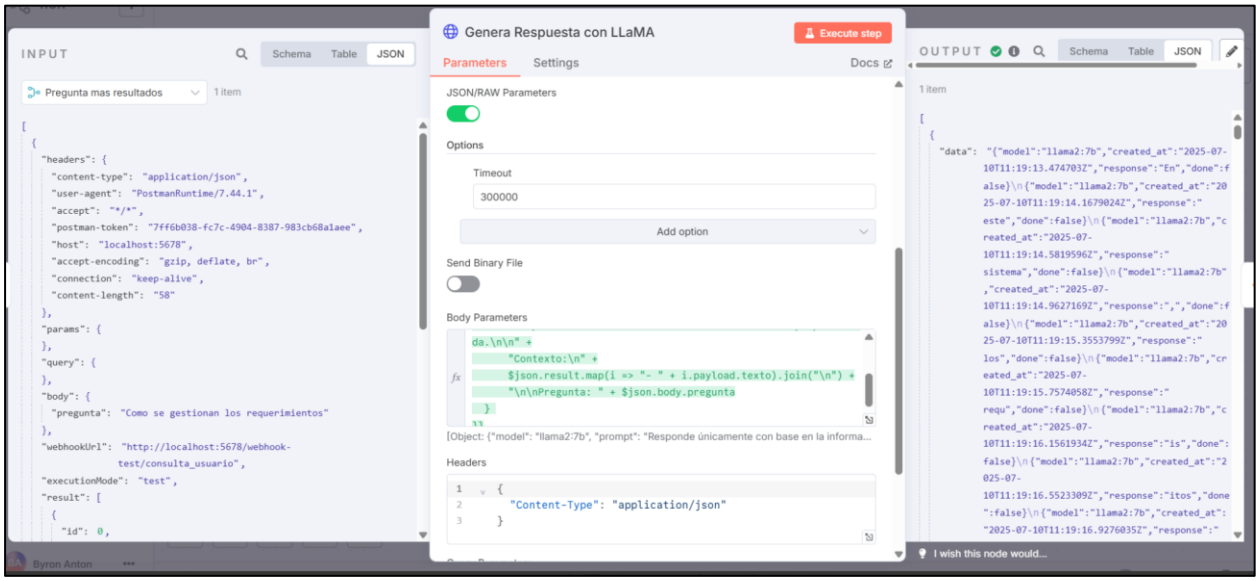


Fig. 26. Configuración del modelo LLaMA con un prompt guiado para generar la respuesta final.

Antes de mostrar la respuesta al usuario, el sistema ejecuta un nodo de limpieza el cual permite filtrar elementos innecesarios y unir adecuadamente los fragmentos generados por el modelo, asegurando que el mensaje final sea claro, sin errores ni repeticiones. De esta forma, se mejora la calidad del texto que el usuario recibe.

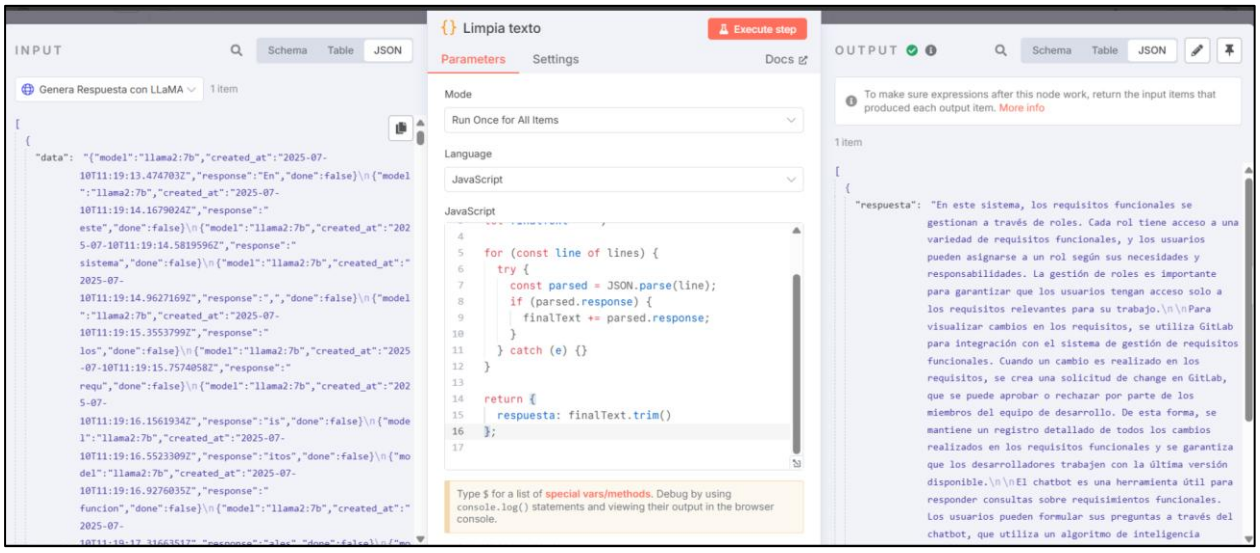


Fig. 27. Proceso de limpieza de texto antes de enviar la respuesta al usuario

Finalmente, el sistema entrega la respuesta al usuario. Este paso ocurre mediante un nodo que toma el mensaje ya limpio y lo envía a través del webhook que originó la consulta. De esta manera, quien hizo la pregunta recibe un texto claro, útil y contextualizado.

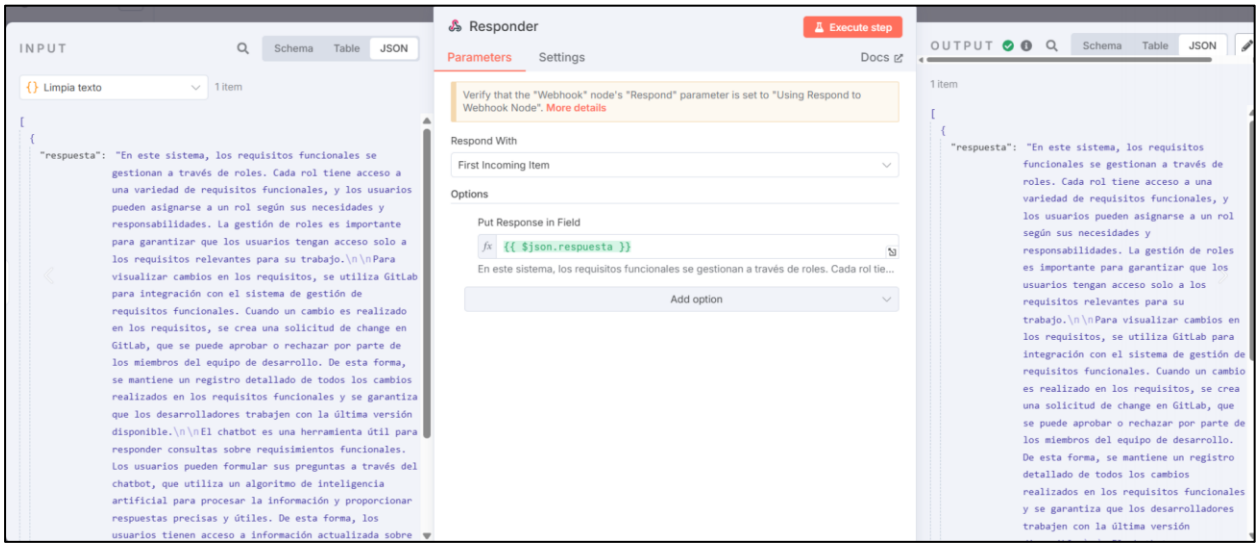


Fig. 28. Nodo de respuesta que entrega el resultado al usuario

La ejecución completa del flujo permitió comprobar que la solución diseñada funciona de forma coherente y entrega respuestas claras en un entorno controlado. Estos resultados confirman que el agente es capaz de procesar una pregunta, buscar información relacionada y generar una respuesta comprensible.

3.3 Comparación entre el modelo actual y el modelo propuesto

3.3.1 Proceso TO BE

Durante el levantamiento de información se evidenciaron limitaciones en el modelo actual de búsqueda y consulta de requerimientos funcionales. Estas limitaciones se traducían en tiempos elevados de búsqueda, ambigüedad en la redacción y falta de trazabilidad de los requerimientos ya utilizados. Por ejemplo, en muchos casos era necesario consultar a otros colaboradores, revisar manualmente proyectos previos y depender del conocimiento individual acumulado.

Con la implementación del nuevo modelo propuesto, estas tareas se automatizaron mediante un agente inteligente que busca, interpreta y entrega información relevante de forma inmediata. Esta transformación se tradujo en una mejora significativa en la eficiencia del proceso, especialmente al reducir el tiempo promedio estimado de consulta de 45 minutos (en el modelo actual) a menos de 7 minutos con el modelo automatizado.

Tabla XII. Comparación general entre AS-IS , TO-BE

Elemento evaluado	Modelo Actual (AS-IS)	Modelo Propuesto (TO-BE)
Búsqueda de requerimientos similares	Manual, dependiente del criterio individual	Automatizada, basada en similitud semántica
Tiempo promedio de respuesta	45 minutos	6 a 7 minutos (ejecución validada)
Fuente de información	Parcial, no centralizada	Integrada y accesible desde una única fuente
Reutilización de información	Baja, con esfuerzo significativo	Alta, basada en vectorización y recuperación semántica
Claridad de información	Variable	Enriquecida con datos contextualizados
Colaboración entre áreas	Limitada, por diferencias en la interpretación	Facilitada por acceso común a datos y lenguaje unificado
Escalabilidad	Baja, no replicable fácilmente a otros procesos	Alta, gracias al diseño modular y reutilizable del agente

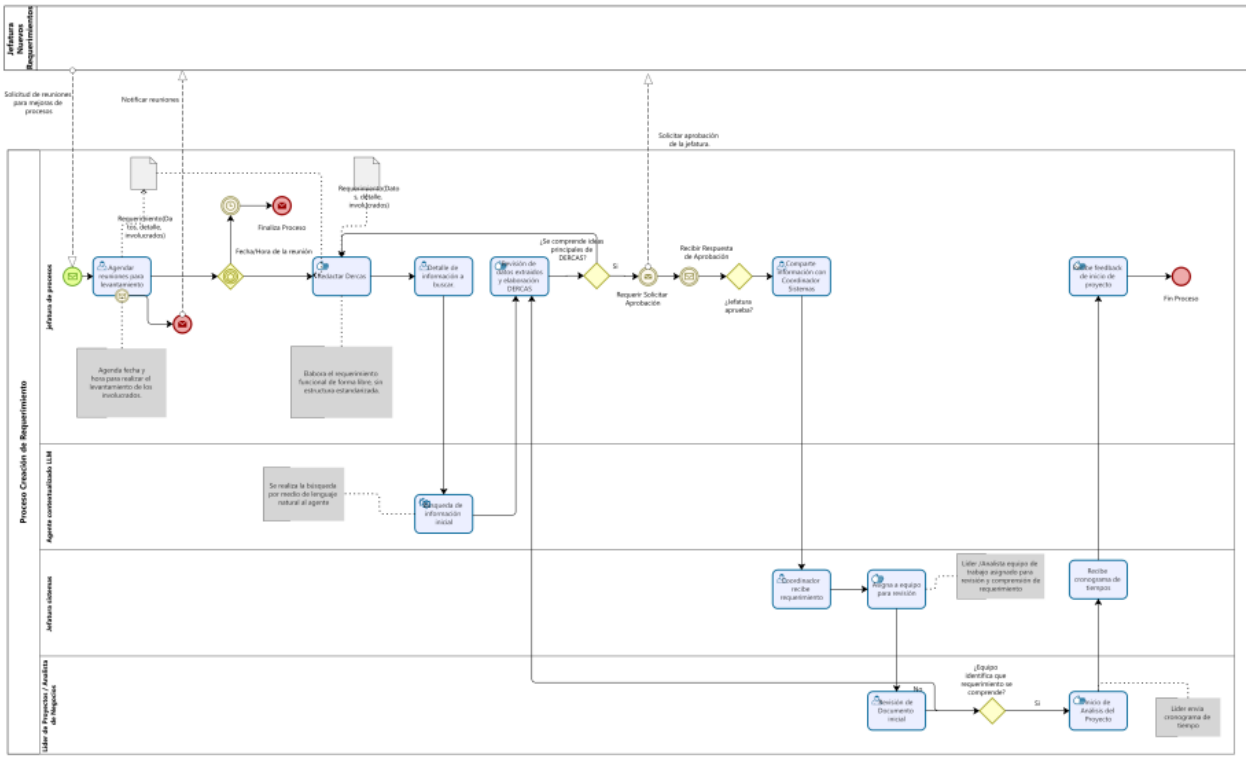


Fig. 29. Diagrama de Proceso TO- BE integración de Agente contextualizado LLM

En conclusión, al comparar el proceso actual con el modelo propuesto, se observa una mejora evidente en la forma en que se consultan los requerimientos. Antes, esta tarea dependía mucho de la memoria o experiencia del equipo, lo que generaba demoras e incluso errores. Con el nuevo diseño, se facilita el acceso a información ya existente y se ahorra tiempo. Estos cambios no solo atienden el problema identificado, sino que demuestran que la solución planteada puede aplicarse también a otros procesos dentro de la empresa.

3.4 Discusión

Los resultados obtenidos evidencian la viabilidad técnica y operativa del agente inteligente de búsqueda diseñado, que integra flujos automatizados en N8N, vectorización con Qdrant y generación de respuestas en lenguaje natural mediante el modelo LLaMA. Esta solución permitió automatizar la recuperación de información desde documentos DERCAS almacenados en GitLab, optimizando así el acceso a requerimientos funcionales históricos.

Durante las pruebas, se observó una reducción significativa en el tiempo de respuesta, pasando de un promedio de 45 minutos a 6-7 minutos, lo cual representa una mejora considerable en la eficiencia operativa del proceso. Este hallazgo cumple con los objetivos, que planteaba diseñar una arquitectura que conecte modelos de lenguaje natural con la información almacenada, para entregar respuestas contextualizadas a los usuarios.
Comparado con el modelo AS-IS, el sistema propuesto (TO-BE) mejoró en múltiples dimensiones:

- La búsqueda de requerimientos similares pasó de ser manual y subjetiva a un proceso automatizado, basado en similitud semántica.
- Se incrementó la claridad y reutilización de la información, gracias a la estandarización de etiquetas, estructuras mínimas y metadatos.
- La solución se diseñó con un enfoque escalable y modular, permitiendo su replicabilidad a otros procesos de la organización.
- El agente esta enfocado para que cualquier usuario pueda hacer uso del mismo, reduciendo la dependencia de personas y el acceso a la información.

Sin embargo, también se identificaron limitaciones relevantes que deben considerarse en futuras iteraciones:

- La calidad y estandarización del contenido en los archivos DERCAS influye directamente en la calidad de la vectorización; documentos mal estructurados generan resultados menos precisos.
- El uso de herramientas como Qdrant y LLaMA requiere una infraestructura robusta, lo cual puede representar una barrera técnica para entornos de bajo presupuesto o con recursos limitados.
- La complejidad en la configuración del flujo N8N, especialmente en la validación de parámetros, manejo de errores y orden de ejecución, puede dificultar su mantenimiento sin personal capacitado.
- Se identificó una oportunidad de mejora en el enriquecimiento del prompt utilizado por el modelo LLaMA, lo que abre camino al uso de técnicas más avanzadas de prompt engineering.

3.5 Conclusiones

El diseño del agente inteligente de búsqueda permitió validar técnicamente la viabilidad de integrar herramientas de automatización N8N, bases vectoriales Qdrant y modelos de lenguaje LLaMA para facilitar el acceso a requerimientos documentados en entornos reales de desarrollo de software.
Entre los principales hallazgos se destacan:

- Se logró vectorizar exitosamente información extraída desde GitLab (issues y archivos DERCAS), lo que permitió almacenarla en una base vectorial semántica consultable.
- La arquitectura diseñada permitió realizar búsquedas en lenguaje natural, entregando respuestas contextualizadas que se basaban exclusivamente en la información disponible en la base de requerimientos.
- El flujo automatizado en n8n demostró un alto grado de adaptabilidad, permitiendo coordinar la interacción entre módulos sin necesidad de desarrollo adicional complejo.
- Se evidenció que la calidad de las respuestas generadas por el modelo LLaMA depende directamente de la riqueza del contexto proporcionado y de una formulación clara del prompt.

Logros alcanzados

- Se cumplió con éxito cada uno de los objetivos específicos definidos en la tesis, incluyendo la estandarización del uso de GitLab, el diseño del flujo inteligente y la validación de la consulta a través del modelo LLaMA.
- Se estableció una arquitectura funcional que demuestra cómo un sistema RAG (Retrieval-Augmented Generation) puede implementarse de tal forma sin depender de soluciones comerciales externas.
- Se consiguió mantener trazabilidad de cada etapa del flujo, desde la entrada de la pregunta hasta la respuesta devuelta al usuario.
- El diseño permite automatizar tareas repetitivas y reducir la dependencia de búsquedas manuales, generando respuestas contextualizadas que hubieran requerido intervención humana en el proceso tradicional.
- Se demostró que la solución es viable y funcional bajo la infraestructura técnica disponible en la empresa, cumpliendo principios de seguridad de la información acordes con normativas establecidas en la misma.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

Al finalizar el diseño propuesto, se lograron validar los componentes esenciales de una arquitectura de búsqueda inteligente para facilitar el acceso de información a requerimientos funcionales dentro de una empresa de telecomunicaciones. A continuación, se presentan los hallazgos más relevantes y su relación directa con los objetivos específicos del proyecto.

- Se identificaron los principales puntos de dolor en los procesos de búsqueda de requerimientos, especialmente en cuanto a ambigüedad en los documentos, pérdida de información útil y largos tiempos de respuesta, esto se identificó mediante encuestas y análisis cualitativo el cual se evidenció la necesidad de mejorar el acceso a requerimientos históricos tanto para los equipos técnicos como de negocio.
- Se logró estandarizar la forma en la que se registra la información de los proyectos en GitLab, a través de la organización lógica de issues, etiquetas y adjuntos estructurados como los documentos DERCAS**,** permitiendo establecer una base ordenada para su posterior vectorización y búsqueda.
- Se definieron flujos automatizados en N8N que integran GitLab, Qdrant y el modelo de lenguaje Llama, permitiendo responder preguntas en lenguaje natural con información contextual proporcionada del repositorio Gitlab, demostrando que es posible interpretar consultas y generar respuestas coherentes basadas en el contenido almacenado.

4.2 Recomendaciones

Tras culminar esta propuesta de diseño de un agente de búsqueda inteligente, se determinan las siguientes recomendaciones primordiales para futuras adaptaciones de mejora del diseño planteado. Estas recomendaciones consideran tanto las limitaciones identificadas como las oportunidades de ampliación del sistema.

- Aunque el sistema ya responde a preguntas en lenguaje natural, por ejemplo, mediante Postman o desde el flujo automatizado en N8N, su uso actual requiere conocimientos técnicos. Se recomienda desarrollar una interfaz web sencilla y segura que se integre con los sistemas existentes de la empresa, permitiendo que analistas, gestores y técnicos consulten información de requerimientos que necesiten.
- Este diseño fue pensado para ejecutarse en servidores de la empresa, sin embargo, el modelo Llama requiere recursos significativos, especialmente en CPU y memoria RAM. Para garantizar una experiencia fluida en producción.
- En esta fase se trabajó exclusivamente con documentos e issues del repositorio GitLab. Sin embargo, la infraestructura empresarial incluye otras fuentes valiosas como bases de conocimiento. Se recomienda adaptar el flujo para que incorpore y procese automáticamente estos recursos, fortaleciendo la base vectorial con información más completa.
- Se recomienda mejorar la estructura de los prompts enviados al modelo Llama, especialmente para escenarios empresariales en los que la precisión es crítica. Esto puede lograrse mediante instrucciones más estrictas, el uso de delimitadores semánticos, la inclusión ejemplos representativos dentro del mismo mensaje que orienten la respuesta esperada incluso ajustes dinámicos del prompt según el tipo de requerimiento.
- Debido a que el diseño está orientado a integrarse en la infraestructura de la empresa, será necesario incorporar mecanismos de control de acceso, autenticación de usuarios y trazabilidad de consultas. Estos aspectos no fueron cubiertos en esta propuesta, pero deben ser considerados para tener un mejor control a la información expuesta en este diseño.

Referencias

[1] S. Amer-Yahia, «Intelligent Agents for Data Exploration», *Proc VLDB Endow*, vol. 17, n.º 12, pp. 4521-4530, ago. 2024, doi: 10.14778/3685800.3685913.

[2] Y. Zheng, Y. Chen, B. Qian, X. Shi, Y. Shu, y J. Chen, «A Review on Edge Large Language Models: Design, Execution, and Applications», *ACM Comput Surv*, vol. 57, n.º 8, mar. 2025, doi: 10.1145/3719664.

[3] D. Zha *et al.*, «Data-centric Artificial Intelligence: A Survey», *ACM Comput Surv*, vol. 57, n.º 5, ene. 2025, doi: 10.1145/3711118.

[4] J. Qiu, C. Wang, y Y. He, «Research on application of intelligent agents in the workflow management system», en *Proceedings. 2005 IEEE Networking, Sensing and Control, 2005.*, 2005, pp. 827-830. doi: 10.1109/ICNSC.2005.1461298.

[5] R. B. M. T. Bandara y R. Wickramarachchi, «Developing an AI Readiness Model for Software Project Management: A Thematic Analysis», en *2025 5th International Conference on Advanced Research in Computing (ICARC)*, 2025, pp. 1-6. doi: 10.1109/ICARC64760.2025.10962923.

[6] A. Gounaris, «Towards Automated Performance Optimization of BPMN Business Processes», en *New Trends in Databases and Information Systems*, M. Ivanović, B. Thalheim, B. Catania, K.-D. Schewe, M. Kirikova, P. Šaloun, A. Dahanayake, T. Cerquitelli, E. Baralis, y P. Michiardi, Eds., Cham: Springer International Publishing, 2016, pp. 19-28.

[7] J. Mangler y S. Rinderle-Ma, «Cloud Process Execution Engine: Architecture and Interfaces». 2022. [En línea]. Disponible en: https://arxiv.org/abs/2208.12214

[8] M. Desmond, E. Duesterwald, V. Isahagian, y V. Muthusamy, «A No-Code Low-Code Paradigm for Authoring Business Automations Using Natural Language». 2022. [En línea]. Disponible en: https://arxiv.org/abs/2207.10648

[9] A.-M. Barthe-Delanoë, S. Truptil, F. Bénaben, y H. Pingaud, «Event-driven agility of interoperability during the Run-time of collaborative processes», *Decis. Support Syst.*, vol. 59, pp. 171-179, 2014, doi: https://doi.org/10.1016/j.dss.2013.11.005.

[10] C. Matthies, F. Dobrigkeit, y G. Hesse, «Mining for Process Improvements: Analyzing Software Repositories in Agile Retrospectives», en *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, en ICSEW'20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 189-190. doi: 10.1145/3387940.3392168.

[11] K. Vilkomir y N. Herndon, «Challenges of Automatic Document Processing with Historical Data», en *Proceedings of the 2024 ACM Southeast Conference*, en ACMSE '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 50-59. doi: 10.1145/3603287.3651200.

[12] T. Chakraborti *et al.*, «From Robotic Process Automation to Intelligent Process Automation: Emerging Trends». 2020. [En línea]. Disponible en: https://arxiv.org/abs/2007.13257

[13] M. Dumas *et al.*, «AI-augmented Business Process Management Systems: A Research Manifesto», *ACM Trans. Manag. Inf. Syst.*, vol. 14, n.º 1, pp. 1-19, ene. 2023, doi: 10.1145/3576047.

[14] L. Bariah, Q. Zhao, H. Zou, Y. Tian, F. Bader, y M. Debbah, «Large Generative AI Models for Telecom: The Next Big Thing?», *IEEE Commun. Mag.*, vol. 62, n.º 11, pp. 84-90, 2024, doi: 10.1109/MCOM.001.2300364.

[15] D. P. Wangoo, «Artificial Intelligence Techniques in Software Engineering for Automated Software Reuse and Design», en *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, 2018, pp. 1-4. doi: 10.1109/CCAA.2018.8777584.

[16] C. Baray y K. Wagner, «Where do intelligent agents come from?», *XRDS*, vol. 5, n.º 4, pp. 4-8, jun. 1999, doi: 10.1145/331648.331653.

[17] Y. Guan *et al.*, «Intelligent Agents with LLM-based Process Automation», en *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, en KDD '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 5018-5027. doi: 10.1145/3637528.3671646.

[18] L. Ma *et al.*, «A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge». 2025. [En línea]. Disponible en: https://arxiv.org/abs/2310.11703

[19] Y. A. Malkov y D. A. Yashunin, «Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs», *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, n.º 4, pp. 824-836, 2020, doi: 10.1109/TPAMI.2018.2889473.

[20] J. J. Pan, J. Wang, y G. Li, «Survey of Vector Database Management Systems». 2023. [En línea]. Disponible en: https://arxiv.org/abs/2310.14021

[21] Z. Jing *et al.*, «When Large Language Models Meet Vector Databases: A Survey». 2025. [En línea]. Disponible en: https://arxiv.org/abs/2402.01763

[22] X. Li, M. Li, X. Xia, J. Song, y X. Bu, «Research on the Architecture Design Based on Large Language Model», en *2024 International Conference on Interactive Intelligent Systems and Techniques (IIST)*, 2024, pp. 203-210. doi: 10.1109/IIST62526.2024.00145.

[23] D. Naik, I. Naik, y N. Naik, «Large data begets large data: studying large language models (LLMs) and its history, types, working, benefits and limitations», en *The International Conference on Computing, Communication, Cybersecurity & AI*, Springer, 2024, pp. 293-314.
[24] L. Jiménez Linares, J. A. López-Gómez, J. Á. Martín-Baos, F. P. Romero, y J. Serrano-Guerrero, «ChatGPT: reflexiones sobre la irrupción de la inteligencia artificial generativa en la docencia universitaria», 2023.
[25] V. Liagkou, E. Filiopoulou, G. Fragiadakis, M. Nikolaidou, y C. Michalakelis, «The cost perspective of adopting Large Language Model-as-a-Service», en *2024 IEEE International Conference on Joint Cloud Computing (JCC)*, IEEE, 2024, pp. 80-83.
[26] S. Zhou, Y. Liao, y S. Tang, «Issues on the Logical Foundations of Knowledge Representation for Intelligent Autonomous Agents», en *2012 Second International Conference on Intelligent System Design and Engineering Application*, 2012, pp. 317-320. doi: 10.1109/ISdea.2012.455.
[27] Y. Xia, N. Jazdi, J. Zhang, C. Shah, y M. Weyrich, «Control Industrial Automation System with Large Language Models». 2024. [En línea]. Disponible en: https://arxiv.org/abs/2409.18009
[28] S. S. Dvivedi, V. Vijay, S. L. R. Pujari, S. Lodh, y D. Kumar, «A Comparative Analysis of Large Language Models for Code Documentation Generation». 2024. [En línea]. Disponible en: https://arxiv.org/abs/2312.10349
[29] X. Yu, L. Liu, X. Hu, J. W. Keung, J. Liu, y X. Xia, «Where Are Large Language Models for Code Generation on GitHub?» 2024. [En línea]. Disponible en: https://arxiv.org/abs/2406.19544
[30] L. Xavier, F. Ferreira, R. Brito, y M. T. Valente, «Beyond the Code: Mining Self-Admitted Technical Debt in Issue Tracker Systems». 2020. [En línea]. Disponible en: https://arxiv.org/abs/2003.09418
[31] M. Ben Chaaben, «Software Modeling Assistance with Large Language Models», en *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, en MODELS Companion '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 188-191. doi: 10.1145/3652620.3676877.
[32] G. Izacard y E. Grave, «Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering», *ArXiv Prepr. ArXiv200701282*, 2021, [En línea]. Disponible en: https://arxiv.org/abs/2007.01282
[33] Qdrant Team, «Qdrant Documentation». 2023. [En línea]. Disponible en: https://qdrant.tech/documentation
[34] G. Docs, «Webhook events». 2025. [En línea]. Disponible en: https://docs.gitlab.com/ee/user/project/integrations/webhooks.html
[35] Z. Wang *et al.*, «Faster and Better LLMs via Latency-Aware Test-Time Scaling». 2025. [En línea]. Disponible en: https://arxiv.org/abs/2505.19634