Modelo basado en agentes inteligentes con ia generativa para el análisis de calidad de requisitos de software.

Maigual Gelpud Jeison Alejandro y Zambrano Mesías Harold Denilson

Programa de ingeniería de sistemas, facultad de ingeniería, Universidad CESMAG

Nota del autor

El presente Trabajo de Grado tiene como propósito cumplir el requisito exigido para optar al título de pregrado como ingenieros de sistemas en la Universidad CESMAG. La correspondecia referente a este trabajo debe dirigirse al Programa de ingeniería de sistemas de la Universidad CESMAG. Correo electrónico: ingenieriadesistemas@unicesmag.edu.co

Modelo basado en agentes inteligentes con ia generativa para el análisis de calidad de requisitos de software.

Maigual Gelpud Jeison Alejandro y Zambrano Mesías Harold Denilson

Ingeniería de sistemas, facultad de ingeniería, Universidad CESMAG

Asesora. Yanira Benavidez

10 de agosto 2026

Nota de aceptación

"Va la aceptación definida desde la universidad".

Nota de exclusión

DEDICATORIA

Culminando mis estudios universitarios, soy plenamente consciente de que todo lo conseguido es, en gran parte, gracias a mis padres. Todo esto es el resultado de su esfuerzo incansable, día tras día, por darme la oportunidad de crecer como profesional y como persona. Gracias a ellos por levantarse cada mañana a luchar por mis sueños, que también son los suyos.

A mi padre, John Maigual: gracias totales por ser mi ejemplo a seguir cada día. Gracias por todo el esfuerzo hecho para alcanzar este logro. Hoy, más que nunca, te agradezco por ser quien me impulsó a no rendirme. Recuerdo cuando todo era solo un sueño lejano... y ahora es una realidad. Una realidad que soñamos juntos cada año, cada mes, y ahora solo faltan días. Tú me enseñaste que los sueños se construyen con trabajo y perseverancia.

A mi madre, Adriana Gelpud: madre, gracias por hacerme más fácil todo esto, no solo con tu apoyo material, sino con ese carácter fuerte y noble que hoy me caracteriza. Gracias por todos esos valores que me has inculcado y que me han abierto puertas que jamás imaginé. Gracias por tu amor incondicional, por tu paciencia infinita y por creer en mí incluso cuando yo mismo dudaba.

Este logro es para las dos personas que más amo en el mundo. Personas que me han forjado en el hombre que soy hoy. Hoy, el niño que lloraba porque le tocaba estudiar largas horas, se convierte en ingeniero; y más que eso, en una persona íntegra. Todo eso es gracias a ustedes. Los amo con todo mi corazón, y todos mis logros serán siempre para ustedes.

A mis abuelos, mami Tere, papi Dani y mami Vicky: gracias por sembrar en mí todo el amor que me pudieron brindar. Ustedes son el tesoro más grande que la vida me ha podido dar. Hoy plasmo aquí, en este documento inundado de lágrimas, mis más sinceras y profundas gracias. Y aunque ya no estés aquí conmigo físicamente, mami Vicky, este logro va por ti. No pude llevarte a mi grado, pero estoy seguro de que desde donde estás, estás feliz por mí, y eso me llena el alma.

A mi familia: porque siempre serán los protagonistas de mi vida. Los amo a todos. Gracias por ser, cada uno de ustedes, un pequeño ejemplo de vida con sus propias historias de superación.

A mis amigos y compañeros conocidos en este viaje: a Cristian Hernández, Fernando Yampuesan, Andrés Vallejo y, en especial, a Harold Zambrano... gracias por hacer esta travesía un viaje más fácil, por las risas compartidas, por los ánimos en los momentos difíciles y por convertirse en una segunda familia, y a mi novia Adriana Santacruz gracias por llegar en este momento y hacerlo mejor.

Este título no es solo mío. Es de todos ustedes. Gracias por hacerme quien soy.

Jeison Maigual

DEDICATORIA

Dedico este logro, en primer lugar, a Dios, por ser mi guía constante en cada paso de este camino, por darme la fortaleza en los momentos difíciles y la sabiduría necesaria para seguir adelante. Gracias por iluminar mi vida, por permitirme llegar hasta este punto y por bendecir cada esfuerzo realizado durante este proceso.

A mis padres, Patricia Mesías y Harold Zambrano, quienes han sido el pilar fundamental de mi vida y el motor de cada uno de mis sueños. Gracias por su amor incondicional, por sus sacrificios, por creer en mí incluso cuando yo dudaba, y por brindarme siempre su apoyo y confianza. Todo lo que soy y lo que he logrado es gracias a ustedes. Este logro también les pertenece.

A mis hermanos, Yeinsy Zambrano y Jhon Mesías, por estar siempre presentes, por su compañía, sus palabras de ánimo y por ser parte importante de mi vida. Gracias por compartir conmigo cada momento, por apoyarme en los desafíos y por motivarme a seguir adelante sin rendirme.

A mi novia, Yessica Hernández, por su amor, paciencia y comprensión durante todo este proceso. Gracias por estar a mi lado en cada etapa, por apoyarme en los momentos de estrés y cansancio, y por creer en mí incluso cuando parecía difícil. Tu compañía ha sido fundamental para no rendirme y seguir avanzando.

A mi tía, Eriqueta Mesías, por su apoyo, cariño y acompañamiento a lo largo de este camino, siendo un pilar importante en mi vida y en mi formación personal.

A mi compañero de este proyecto de grado, Jeison Maigual, por su compromiso, esfuerzo y dedicación durante todo el desarrollo de esta investigación. Gracias por el trabajo en equipo, por el apoyo mutuo y por enfrentar juntos cada reto presentado en este proceso. Este logro también es fruto de ese esfuerzo conjunto.

De igual manera, a mis compañeros de carrera, Cristian Hernández y Fernando Yampuezan, quienes han sido parte fundamental de este camino académico. Gracias por su amistad, apoyo y por compartir cada etapa de este proceso, contribuyendo a mi crecimiento personal y profesional. A todos ustedes, gracias por ser mi inspiración, mi apoyo y mi fortaleza. Este logro no es solo mío, es el resultado del amor, el esfuerzo y el acompañamiento de cada uno de ustedes a lo largo de este camino.

Harold Zambrano

AGRADECIMIENTOS

Los autores expresan su más sincero y profundo agradecimiento a los directores Mg. Jorge Albeiro Rivera Rosero y Dr. Hugo Armando Ordóñez Erazo, por su valiosa orientación académica, acompañamiento permanente y compromiso durante todo el proceso investigativo. Sus aportes, observaciones y recomendaciones fueron fundamentales para fortalecer el enfoque, la calidad y la rigurosidad de este trabajo, contribuyendo significativamente a la consolidación de los resultados alcanzados.

Asimismo, se reconoce y agradece al Programa de Ingeniería de Sistemas y a la Universidad CESMAG, por brindar un entorno académico propicio para el desarrollo de la investigación, así como por fomentar el pensamiento crítico, la innovación y la aplicación del conocimiento en contextos reales. El apoyo institucional y la formación recibida fueron clave para la formulación, implementación y validación de la propuesta presentada.

De igual manera, se extiende un agradecimiento especial a los estudiantes que participaron en la fase de validación del modelo, quienes, con su disposición y compromiso, hicieron posible la recolección de datos y la evaluación de la herramienta en un entorno académico controlado. Su participación fue fundamental para evidenciar la aplicabilidad y efectividad de la propuesta.

Finalmente, los autores agradecen a todas las personas que, de una u otra forma, contribuyeron al desarrollo de esta investigación, ya sea a través de su apoyo académico, técnico o motivacional, permitiendo culminar satisfactoriamente este proceso.

TABLA DE CONTENIDO

Pág.

GLOSARIO....12
RESUMEN....13
ABSTRACT....14
INTRODUCCIÓN....15
I. PROBLEMA DE INVESTIGACIÓN....16
A. OBJETO O TEMA DE ESTUDIO....16
B. LÍNEA DE INVESTIGACIÓN....16
C. PLANTEAMIENTO DEL PROBLEMA....16
D. FORMULACIÓN DEL PROBLEMA....17
E. OBJETIVOS....18
F. JUSTIFICACIÓN....18
G. DELIMITACIÓN....20
II. MARCO TEÓRICO....21
A. ANTECEDENTES....21
H. SUPUESTOS TEÓRICOS....25
I. VARIABLES DEL ESTUDIO....30
J. FORMULACIÓN DE HIPÓTESIS....32
III. METODOLOGÍA....33
A. PARADIGMA....33
B. ENFOQUE....33
C. MÉTODO....33
D. TIPO DE INVESTIGACIÓN....33
K. DISEÑO DE LA INVESTIGACIÓN....34
L. POBLACIÓN....36
M. MUESTRA....37
N. TÉCNICAS DE RECOLECCIÓN DE INFORMACIÓN....38
O. INSTRUMENTO DE RECOLECCIÓN DE DATOS....38
P. DESARROLLO METODOLÓGICO....39
IV. RESULTADOS DE LA INVESTIGACIÓN....51
A. OBJETIVO ESPECÍFICO 1....51
B. OBJETIVO ESPECÍFICO 2....64

C. OBJETIVO ESPECÍFICO 3 67
V. ANÁLISIS Y DISCUSIÓN DE LOS RESULTADOS 68
CONCLUSIONES 71
RECOMENDACIONES 72
REFERENCIAS BIBLIOGRÁFICAS 73
ANEXOS 81

LISTA DE FIGURAS

Fig 1. Inteligencia artificial (IA) 26
Fig 2. Agentes inteligentes 27
Fig 3. Procesamiento de lenguaje natural 28
Fig 4. Requisitos funcionales 29
Fig. 5. Fases del desarrollo metodológico. 40
Fig. 6. Proceso de consolidación de atributos. 42
Fig. 7. Arquitectura del modelo RE-AGENTS 61
Fig. 8. Ejemplo de prompt del agente Analista de Requisitos 62
Fig 9. Ejemplo de salida generada por el sistema 63
Fig. 10. Interfaz principal de la herramienta web. 65
Fig. 11. Módulo de gestión de proyectos. 65
Fig. 12. Módulo de gestión de requisitos. 66
Fig. 13. Resultado del análisis de requisitos. 67

LISTA DE TABLAS

Pag.

TABLA I. Metodología Preliminar 34
TABLA II. Consolidación de atributos de calidad identificados en la literatura. 51
TABLA III. Matriz de caracterización de los atributos de calidad de requisitos. 56
TABLA IV. Comparación entre modelos LLM actuales 58
TABLA V. Comparación cuantitativa de modelos LLM mediante métricas de evaluación 59
TABLA VI. Resultados de la prueba de validación 67

LISTA DE ANEXOS

A. Instrumentos de validación 81
B. Casos de estudio 81
C. Evidencia de la herramienta 82
D. Código o repositorio 82
E. Encuesta usabilidad herramienta 82

GLOSARIO

AGENTE INTELIGENTE: Sistema autónomo que percibe información de su entorno, la procesa y ejecuta acciones orientadas al cumplimiento de un objetivo específico.

AMBIGÜEDAD: Presencia de términos o expresiones que admiten más de una interpretación y afectan la claridad de un requisito de software.

CALIDAD DE REQUISITOS DE SOFTWARE: Grado en que los requisitos cumplen atributos como claridad, consistencia, completitud, verificabilidad y trazabilidad.

COMPLETITUD: Característica que indica que el requisito contiene la información necesaria para su comprensión, validación e implementación.

CONSISTENCIA: Propiedad mediante la cual los requisitos no presentan contradicciones entre sí ni con los objetivos del sistema.

IA GENERATIVA: Rama de la inteligencia artificial capaz de producir contenido nuevo, como texto y sugerencias, a partir de patrones aprendidos.

INGENIERÍA DE REQUISITOS: Disciplina encargada de identificar, analizar, documentar, validar y gestionar los requisitos de un sistema de software.

LLM: Sigla de Large Language Model, utilizada para referirse a modelos de lenguaje de gran escala entrenados para comprender y generar lenguaje natural.

PLN: Sigla de Procesamiento de Lenguaje Natural, área que permite a las máquinas interpretar, analizar y generar lenguaje humano.

REQUISITO FUNCIONAL: Especificación que describe una función o servicio que el sistema debe ejecutar para satisfacer una necesidad del usuario.

TRAZABILIDAD: Capacidad de seguir el origen, evolución y relación de un requisito a lo largo del ciclo de vida del software.

VALIDACIÓN SEMÁNTICA: Evaluación del significado e interpretación de los requisitos para detectar inconsistencias, ambigüedades o errores de intención.

RESUMEN

La calidad de los requisitos funcionales constituyó un factor crítico en el éxito de los proyectos de software; sin embargo, su validación manual solía ser un proceso subjetivo, costoso y propenso a omisiones. En este contexto, la presente investigación tuvo como objetivo general proponer y evaluar un modelo basado en agentes inteligentes que integró inteligencia artificial generativa para el análisis automatizado de la calidad de requisitos. El estudio se desarrolló bajo un paradigma cuantitativo, con enfoque aplicado y método experimental, incluyendo pruebas piloto que compararon el desempeño del análisis tradicional frente a una herramienta web sustentada en un modelo multiagente denominado RE-AGENTS.

La propuesta articuló técnicas de procesamiento de lenguaje natural y modelos de lenguaje de gran escala, junto con agentes especializados que desempeñaron roles específicos para identificar defectos como ambigüedades, inconsistencias, incompletitudes y falta de verificabilidad en la redacción de requisitos. Como principales resultados, se logró consolidar una matriz de atributos de calidad y desarrollar una herramienta funcional que evidenció una reducción significativa en el tiempo de análisis, así como un incremento en la detección de errores en comparación con la validación manual.

Se concluyó que la integración de IA generativa y arquitecturas multiagente constituyó una alternativa viable y eficiente para fortalecer los procesos de ingeniería de requisitos, especialmente en entornos académicos. Asimismo, el estudio aportó un modelo replicable con potencial de escalabilidad hacia contextos organizacionales, contribuyendo a mejorar la calidad del software desde sus etapas iniciales.

Palabras clave: Agentes Inteligentes, Calidad de Requisitos, IA Generativa, Ingeniería de Requisitos, LLM, Procesamiento de Lenguaje Natural

ABSTRACT

The quality of functional requirements was a critical factor in the success of software projects; however, manual validation was often subjective, time-consuming, and prone to omissions. In this context, this research aimed to propose and evaluate a model based on intelligent agents integrating generative artificial intelligence for the automated analysis of requirements quality. The study followed a quantitative paradigm, with an applied approach and experimental method, including pilot tests comparing traditional validation with a web-based tool supported by a multi-agent model called RE-AGENTS.

The proposal combined natural language processing techniques and large language models with specialized agents performing specific analytical roles to detect defects such as ambiguities, inconsistencies, incompleteness, and lack of verifiability in requirement statements. The main results include the consolidation of a quality attribute matrix and the development of a functional tool that significantly reduced analysis time while increasing error detection compared to manual methods.

It was concluded that the integration of generative AI and multi-agent architectures represented a viable and efficient alternative to strengthen requirements engineering processes, particularly in academic settings. Additionally, this study provided a replicable model with scalability potential for organizational environments, contributing to improving software quality from its initial stages.

Keywords: Generative AI, Intelligent Agents, Large Language Models, Natural Language Processing, Requirements Engineering, Requirements Quality

INTRODUCCIÓN

En el ámbito del desarrollo de software, la calidad de los requisitos fue un factor crucial que impactó directamente en el éxito de cualquier proyecto. Los requisitos mal formulados, ambiguos o incompletos provocaron errores costosos, demoras en el desarrollo y fallos en la funcionalidad, afectando negativamente tanto al equipo de desarrollo como al cliente y al usuario final [1]. Estos desafíos impulsaron un creciente interés en la integración de técnicas avanzadas de inteligencia artificial (IA), especialmente aquellas orientadas a mejorar el análisis y la validación de requisitos desde las etapas iniciales del ciclo de vida del software.

Esta investigación tuvo como propósito proponer un modelo que combinó agentes inteligentes con inteligencia artificial generativa para automatizar el análisis de calidad de los requisitos funcionales del software. El modelo fue diseñado para detectar errores comunes como ambigüedades, inconsistencias y contradicciones, mediante el uso de técnicas de procesamiento de lenguaje natural (PLN) y modelos de lenguaje de gran escala (LLMs).

La solución se implementó en forma de una plataforma web interactiva, cuyo objetivo no fue participar en todas las etapas del desarrollo del software, sino facilitar el proceso de validación y análisis de requisitos en fases tempranas. A través de la interacción entre los usuarios y los agentes inteligentes, se logró mejorar la calidad de los requisitos antes de que avanzara el desarrollo técnico, contribuyendo así a reducir errores posteriores, mejorar la toma de decisiones y optimizar los tiempos de desarrollo.

Esta propuesta, por tanto, no garantizó por sí sola la calidad del producto final, pero sí permitió incrementar las probabilidades de éxito del proyecto al intervenir en la etapa crítica de definición de requisitos, que fue donde comúnmente se originaron muchos de los problemas que afectaron la calidad global del software. De este modo, el proyecto realizó una contribución significativa al campo de la ingeniería de requisitos, al integrar herramientas de vanguardia que combinaron la capacidad adaptativa de la IA generativa con la autonomía colaborativa de los agentes inteligentes.

I. PROBLEMA DE INVESTIGACIÓN

A. OBJETO O TEMA DE ESTUDIO

Calidad de requisitos funcionales de software.

B. LÍNEA DE INVESTIGACIÓN

La investigación se centró en el campo de la ingeniería de software, principalmente en mejorar la calidad de los requisitos mediante la aplicación de la inteligencia artificial (IA) y las metodologías automatizadas, aprovechando el procesamiento del lenguaje natural (NLP), el aprendizaje automático (ML) y los sistemas basados en agentes inteligentes [2]. Además, la investigación se esforzó por automatizar la detección de inconsistencias, ambigüedades, omisiones y redundancias dentro de los requisitos de software, lo cual facilitó la elaboración de especificaciones más precisas desde las fases preliminares del ciclo de vida del desarrollo del software. Esta investigación estuvo alineada con las tendencias de la ingeniería de software de ese momento, que promovieron el uso de tecnologías inteligentes para optimizar procesos, minimizar errores humanos y garantizar proyectos más robustos y acordes con las necesidades de los clientes [3].

SUBLÍNEA DE INVESTIGACIÓN

Ingeniería de Requisitos.

Esta propuesta se alineó con metodologías modernas para el análisis, especificación, validación y gestión de requisitos de software, integrando tecnologías avanzadas como la IA generativa, el procesamiento de lenguaje natural (PLN) y los sistemas multiagente [4]. El objetivo principal fue mejorar la calidad y eficiencia en la elaboración de requisitos mediante modelos de IA capaces de interpretar y generar documentación de forma automática o semiautomática. Además, se incorporaron métodos de validación que aseguraron la coherencia, claridad, completitud y viabilidad de los requisitos [5], así como mecanismos colaborativos basados en agentes inteligentes que facilitaron la negociación, priorización y resolución de conflictos de manera dinámica y adaptativa [6]. Esta integración buscó optimizar el ciclo de vida del software y reducir errores en fases posteriores del desarrollo.

C. PLANTEAMIENTO DEL PROBLEMA

En los proyectos de desarrollo de software, la calidad de los requisitos fue un factor crucial para garantizar un buen producto [7]. Sin embargo, se encontraron dificultades como la ambigüedad, incompletitud, contradicciones, cambios constantes y falta de priorización que impactaron negativamente tanto en el proceso de desarrollo como en el producto final [8].

La baja calidad del software a menudo se asoció con la diversidad de perspectivas entre el cliente y los diferentes stakeholders (partes interesadas) en el desarrollo del proyecto [9]. Uno de los errores más comunes fue que los desarrolladores no distinguieron adecuadamente entre las actividades de investigación y desarrollo, lo que creó ambigüedades en la definición de requisitos y afectó directamente la calidad del producto final [10].

Todos estos problemas y descuidos hicieron que los productos de software se enfrentaran a situaciones que los afectaron en diferentes aspectos, como tiempos de entrega tardíos, costos elevados, insatisfacción del cliente e incluso daños a la reputación de los desarrolladores [11]. Por lo cual se pudo deducir que los puntos anteriormente nombrados estuvieron directamente asociados a requisitos mal definidos, cambios repentinos y demás situaciones que dificultaron un buen desarrollo y condujeron a que el producto final tuviera una calidad deficiente.

En este sentido, la problemática que se abordó en este proyecto se centró en la baja calidad de los requisitos de software, entendida desde dos esferas: por un lado, la calidad en la definición de los requisitos, y por otro, la claridad en su redacción, que debió evitar imprecisiones, inconsistencias y ambigüedades, y facilitar su comprensión [12].

Los desafíos para los desarrolladores de software radicaron tanto en la recolección precisa de información como en la comunicación con los stakeholders (partes interesadas). Por otro lado, los clientes y usuarios también se enfrentaron a dificultades para comunicar sus necesidades de forma clara y consistente, lo cual afectó la precisión de los requisitos [13].

D. FORMULACIÓN DEL PROBLEMA

¿Cómo puede la inteligencia artificial generativa, apoyada en agentes inteligentes, contribuir a mejorar la calidad de los requisitos de software, tanto en su definición como en su claridad, mediante un análisis automatizado que permita reducir errores tempranos, ambigüedades e incompletitudes, a través de iteraciones continuas de verificación, refinamiento y retroalimentación?

E. OBJETIVOS

1) Objetivo general

Proponer un modelo con IA generativa para el análisis automatizado de la calidad de los requisitos de software, con el fin de detectar errores tempranos y asegurar especificaciones bien estructuradas y alineadas con las necesidades del cliente o usuario final.

2) Objetivos específicos

- Caracterizar los atributos esenciales de un requisito de software, con el fin de establecer las bases para la definición de un modelo de análisis automatizado mediante inteligencia artificial generativa, que integre los componentes necesarios para garantizar su efectividad y aplicabilidad.

- Desarrollar una herramienta software basada en el modelo definido previamente, integrando agentes inteligentes para procesar y analizar requisitos funcionales de manera automatizada, que asista a los desarrolladores en la identificación y detección de inconsistencias y errores en los requisitos funcionales.

- Aplicar un método de validación con usuarios, que permita evaluar la efectividad del modelo y la herramienta desarrollada, mediante métricas como la precisión en la detección de errores y el tiempo requerido para la validación de requisitos en proyectos académicos.

F. JUSTIFICACIÓN

La calidad deficiente de los requisitos en los proyectos de desarrollo de software siguió siendo un reto primordial tanto para la industria como para la academia [14]. Diversos estudios evidenciaron que entre el 40% y el 60% de los defectos en software se originaron en la fase de requisitos, lo que lo convirtió en una de las principales causas de fallos en los proyectos [59], [61]. Además, corregir un error en etapas tardías del desarrollo pudo costar entre 10 y 100 veces más que solucionarlo en fases tempranas, según reportes ampliamente citados en la industria como los del Systems Sciences Institute y estudios de IBM [60].

Problemas como la ambigüedad, contradicciones, incompletitud y falta de priorización impactaron directamente en el desarrollo de los proyectos, generando sobrecostos, retrasos en la entrega e insatisfacción del cliente al no cumplirse con sus necesidades [15]. Incluso, se estimó que los proyectos con requisitos deficientes pudieron incrementar sus costos totales hasta en un 30% o

más, afectando la eficiencia organizacional y la calidad del producto final [62]. Frente a este panorama, resultó necesaria la búsqueda de soluciones innovadoras que permitieran mejorar la calidad de los requisitos desde las etapas iniciales del ciclo de vida del software.

En este contexto, la incorporación de herramientas basadas en IA generativa, como los modelos de lenguaje de gran escala (LLMs) —por ejemplo, ChatGPT, DeepSeek, Gemini, entre otros—, en conjunto con agentes inteligentes [16], abrió nuevas posibilidades para automatizar tareas complejas dentro de la ingeniería de software. Estos modelos demostraron una alta capacidad para comprender lenguaje natural, identificar inconsistencias y proponer mejoras en textos técnicos, lo que los convirtió en una alternativa prometedora para el análisis de requisitos.

Particularmente, modelos como MARE (Multi-Agent Requirements Engineering) evidenciaron el potencial de esta sinergia, al distribuir tareas como la elicitación, modelado, verificación y especificación entre agentes inteligentes asistidos por IA generativa [17]. Este tipo de enfoques permitió no solo automatizar procesos, sino también enriquecer el análisis al incorporar múltiples perspectivas de evaluación sobre un mismo requisito, mejorando así su calidad de forma integral.

A partir de estas bases, el presente proyecto se inspiró en enfoques similares, pero propuso el diseño de un modelo propio que, sin necesidad de desarrollar un LLM desde cero, se apoyó en estas tecnologías para asistir el análisis de la calidad de los requisitos funcionales en proyectos de software. La propuesta buscó aprovechar las capacidades de los agentes inteligentes para identificar automáticamente problemas comunes como ambigüedad, contradicciones y falta de claridad en los requisitos [18], además de generar sugerencias de mejora de manera automática.

Desde un enfoque teórico, el proyecto contribuyó a la ingeniería de requisitos al integrar tecnologías emergentes dentro de un modelo estructurado y coherente. Metodológicamente, propuso una arquitectura adaptable tanto en contextos académicos como empresariales. En el plano práctico, se esperaba que el modelo optimizara el proceso de validación de requisitos, reduciendo tiempos de análisis y aumentando la detección de errores, lo cual contribuyó a disminuir costos asociados a fallos en etapas posteriores del desarrollo.

En el ámbito académico, este trabajo fortaleció la formación de competencias en inteligencia artificial generativa y desarrollo de software, permitiendo a los estudiantes interactuar con herramientas innovadoras. Finalmente, en términos sociales y económicos, el modelo presentó un

alto potencial de aplicación en entornos reales, facilitando la creación de proyectos más eficientes, sostenibles y alineados con las necesidades del usuario, al reducir retrabajos, costos de mantenimiento y riesgos en el desarrollo de software.

G. DELIMITACIÓN

Este proyecto de investigación se enfocó exclusivamente en el análisis de la calidad de los requisitos funcionales en el desarrollo de software, dejando fuera otras fases del ciclo de vida, como el diseño, la implementación o el mantenimiento [23]. Su objetivo principal fue evaluar y mejorar aspectos clave de los requisitos, tales como la claridad, coherencia y completitud, sin abordar validaciones formales o legales en contextos altamente regulados.

El enfoque metodológico se basó en la utilización de modelos de lenguaje de gran escala (LLMs) y agentes inteligentes como herramientas de apoyo para identificar deficiencias comunes en los requisitos, tales como ambigüedades, contradicciones y ausencia de criterios de priorización [24].

Aunque el modelo se diseñó inicialmente para ser implementado en contextos académicos o pedagógicos, también se proyectó su aplicación en entornos empresariales, especialmente en organizaciones que buscaban automatizar y optimizar sus procesos de recolección y análisis de requisitos. En su fase inicial, el modelo estuvo enfocado en proyectos desarrollados en idioma español, por lo que se previó la necesidad de ajustes futuros para su adaptación a otros idiomas y contextos culturales.

II. MARCO TEÓRICO

A. ANTECEDENTES

INTERNACIONAL

La ingeniería de requisitos evolucionó considerablemente durante la última década, impulsada por la creciente necesidad de desarrollar software de alta calidad…La ingeniería de requisitos evolucionó considerablemente durante la última década, impulsada por la creciente necesidad de desarrollar software de alta calidad desde las fases tempranas del ciclo de vida. Diversos estudios evidenciaron que los errores en los requisitos funcionales constituyeron una de las causas principales de fallos en proyectos de software, representando hasta el 50% de los errores detectados en etapas posteriores del desarrollo [10]. Esto motivó una transformación en las metodologías de análisis, adoptando tecnologías avanzadas como la inteligencia artificial (IA), el procesamiento de lenguaje natural (NLP), los modelos de lenguaje a gran escala (LLM) y los sistemas multiagente [17], [25].

A escala internacional, Henning Femmer, Michael Unterkalmsteiner y Tony Gorschek (2023) realizaron una evaluación sistemática de las reglas de calidad aplicables a artefactos de requisitos. Su investigación demostró que más del 50% de los defectos pudieron ser detectados mediante herramientas automatizadas basadas en reglas, pero que aún persistió una fuerte dependencia de las revisiones manuales, lo que elevó los costos y prolongó los tiempos de entrega [26]. Este antecedente sustentó la necesidad de avanzar hacia una mayor automatización en la evaluación de requisitos, lo cual fue un eje central en esta investigación, que buscó reemplazar el enfoque manual con un modelo inteligente basado en agentes autónomos.

Por su parte, Veizaga, Shin y Briand (2023) desarrollaron la herramienta Paska, orientada a la identificación de "smells" en requisitos escritos en lenguaje natural. Esta herramienta semiautomática empleó técnicas de procesamiento de lenguaje natural (NLP) y lenguajes controlados, alcanzando precisiones superiores al 89% en entornos industriales reales [27]. Sin embargo, su capacidad fue limitada en contextos altamente técnicos o específicos del dominio. Esta limitación fue abordada en la presente investigación mediante una arquitectura colaborativa e iterativa de agentes inteligentes con IA generativa, que permitió una mayor adaptación y comprensión contextual de los requisitos.

Una contribución más reciente fue la de Rasheed, Nguyen Duc, Systä y Abrahamsson (2025), quienes investigaron la integración de agentes autónomos con modelos de lenguaje a gran escala (LLM) para asistir en la generación, validación y análisis de requisitos de software [28]. Su propuesta representó un cambio de paradigma en la ingeniería de requisitos al incorporar sistemas que no solo ejecutaron tareas, sino que aprendieron y se adaptaron dinámicamente al entorno. Este enfoque fue fundamental para el modelo propuesto en esta investigación, que empleó agentes inteligentes con técnicas de Input Engineering para permitir una validación continua y adaptativa de los requisitos frente a escenarios complejos y cambiantes.

Asimismo, Jin, Chen y Wang (2024) introdujeron un modelo colaborativo multiagente denominado MARE (Multi-Agent Collaboration Model for Requirements Engineering), basado en LLMs. Este modelo dividió las tareas de la ingeniería de requisitos entre diferentes agentes —como obtención, verificación y especificación—, lo cual incrementó la eficiencia y precisión del proceso. Según sus resultados empíricos, MARE superó a otras metodologías anteriores en un 15.4% en cuanto a calidad de los modelos generados [17]. Este antecedente validó empíricamente la capacidad de los sistemas multiagente potenciados por IA generativa para optimizar de forma integral el ciclo de vida de los requisitos, lo que se alineó directamente con los objetivos de este estudio.

Finalmente, Zhang et al. (2025) propusieron RE4NLP, una arquitectura inteligente que combinó aprendizaje profundo y modelos de lenguaje para evaluar automáticamente la calidad sintáctica y semántica de los requisitos funcionales. Los experimentos demostraron mejoras significativas en precisión y recall en escenarios reales de desarrollo de software [29]. Este trabajo aportó evidencia sobre la efectividad del uso de NLP y redes neuronales en el análisis de calidad de requisitos, lo cual fortaleció la base técnica del modelo propuesto en esta investigación, orientado a maximizar la precisión del análisis automatizado mediante agentes inteligentes y técnicas generativas.

NACIONALES

En Colombia, diversos estudios habían abordado la problemática de la calidad de los requisitos de software, especialmente en contextos de pequeñas y medianas empresas, lo cual resultó relevante para el fortalecimiento de modelos innovadores que integraron tecnologías emergentes como la inteligencia artificial (IA) y los sistemas multiagente.

Uno de estos aportes provino de León Torrecilla (2023), quien propuso una estrategia ágil para garantizar la calidad en proyectos de desarrollo de software en Pymes y Mipymes de Bogotá, aplicando metodologías como Scrum y Kanban. Su enfoque se centró en las fases críticas del ciclo de vida del software, identificando fallas frecuentes en los requisitos, pruebas y mantenimiento [30]. Este estudio destacó la importancia de mejorar la calidad desde etapas tempranas y con recursos limitados, lo que proporcionó una base contextual sólida para adaptar soluciones tecnológicas innovadoras, como el uso de agentes inteligentes e IA generativa, en entornos empresariales similares.

Por su parte, Navas Triana y Quintana Paternina (2021) desarrollaron un programa de pruebas basado en las reglas IEEE 830 para mejorar el desarrollo de software en la empresa TICS S.A.S. Identificaron falencias en la organización de los proyectos, en la claridad de los requerimientos y en la ausencia de una hoja de ruta estructurada, lo cual afectó negativamente la satisfacción del cliente. Su propuesta consistió en implementar un sistema de gestión documental para administrar y revisar los requisitos [31]. Estos hallazgos resaltaron la necesidad de herramientas que permitieran mejorar la organización, trazabilidad y almacenamiento de la información de requisitos, un objetivo que pudo ser abordado con soluciones automatizadas basadas en agentes inteligentes.

Asimismo, Ubarnes Martínez y Gómez Sánchez (2025) presentaron un enfoque basado en técnicas de procesamiento de lenguaje natural (PLN) para generar esquemas preconceptuales a partir de historias de usuario, permitiendo una representación más clara del comportamiento del software. A diferencia de modelos guiados, su propuesta trabajó con entradas naturales y buscó generar diagramas interpretables por los equipos de desarrollo [32]. Este enfoque evidenció el potencial del PLN para mejorar la interpretación de los requisitos del usuario, lo cual se complementó con la propuesta de esta investigación al integrar agentes inteligentes que analizaron, transformaron y refinaron automáticamente los requisitos en lenguaje natural, utilizando técnicas de ingeniería de Prompt e interacción con modelos generativos.

En otra contribución, Víctor Daniel Gil-Vera y Cristian Seguro-Gallego (2022) desarrollaron una aplicación distribuida que empleó aprendizaje automático y bases de conocimiento para detectar ambigüedades en la especificación de requisitos. Este desarrollo, llevado a cabo como parte de una pasantía en Eprocess S.A.S., se enfocó en la resolución automática de ambigüedades, logrando una

herramienta funcional que apoyó al equipo de desarrollo en la comprensión de los requisitos [33]. Este trabajo demostró la viabilidad de aplicar técnicas de inteligencia artificial para mejorar el análisis de requisitos, lo cual se amplió en la presente investigación mediante un enfoque colaborativo entre agentes inteligentes que no solo identificaron errores, sino que también dialogaron entre sí para proponer correcciones y validaciones.

Finalmente, Peláez, Cohuó, Delgado, Toro y Arias (2022) desarrollaron el modelo CHAMÍ, una propuesta para asegurar la calidad de los requerimientos funcionales y no funcionales en MiPymes de la industria del software, especialmente en la región del Eje Cafetero. Su modelo fue validado con la participación de expertos académicos e industriales, evidenciando que fue posible fortalecer la competitividad de estas empresas mediante procesos sistemáticos de análisis de requisitos [34]. Este modelo representó un precedente importante, ya que demostró la necesidad y eficacia de soluciones especializadas para este tipo de organizaciones, alineándose con la propuesta de utilizar modelos inteligentes que automatizaran, adaptaran y optimizaran la gestión de los requisitos de software en escenarios con recursos limitados.

REGIONALES

En el ámbito regional, se realizaron esfuerzos significativos para fortalecer las capacidades en desarrollo de software, lo que contribuyó directamente al mejoramiento del análisis y gestión de requisitos desde la formación académica y técnica.

En Nariño, la automatización de pruebas en el desarrollo ágil de software se presentó como una pieza muy importante para asegurar la calidad del software. Calpa Bravo y Garzón Mora (2022), para su tesis del programa de Ingeniería de Sistemas de la Universidad de Nariño, propusieron un proceso para el desarrollo de pruebas unitarias y componentes automatizados, todo esto con el fin de garantizar una correcta calidad. Este estudio caracterizó el uso de pruebas en artículos de 2010 a 2020 y mostró un bajo uso de pruebas unitarias y una falta de pruebas de componentes. El proceso propuesto utilizó herramientas y artefactos de automatización y gestión de no conformidades para especificar historias de usuario, escenarios y casos de prueba [35].

La Universidad de Nariño (2023) promovió la formación especializada en ingeniería de software a través de su programa de Especialización en Construcción de Software, un posgrado presencial de dos semestres que fue recientemente acreditado con Registro Calificado por el Ministerio de

Educación Nacional. Este programa fortaleció competencias clave como la gestión de requisitos, metodologías ágiles y aseguramiento de la calidad, orientando a los profesionales hacia la optimización de procesos empresariales mediante soluciones tecnológicas eficientes [36]. Este tipo de formación resultó fundamental para preparar talento humano capacitado en el análisis riguroso y estructurado de requisitos, alineado con los objetivos de investigaciones que propusieron soluciones inteligentes y automatizadas en el ámbito del desarrollo de software.

Por otro lado, ParqueSoft Nariño y el SENA (2025) implementaron programas de formación técnica en análisis y desarrollo de software con una duración de 27 meses, que combinaron teoría y práctica para ofrecer habilidades en todas las fases del ciclo de vida del software. Desde la recopilación de requisitos hasta el mantenimiento, este programa proporcionó a los estudiantes competencias prácticas esenciales para enfrentar los desafíos reales del desarrollo de software [37]. Esta iniciativa fue especialmente relevante al mostrar cómo la formación técnica pudo integrarse con procesos formales de análisis y desarrollo, lo cual fue coherente con propuestas que involucraron agentes inteligentes capaces de asistir en tareas como la validación y refinamiento de requisitos.

Asimismo, la Corporación Universitaria Autónoma de Nariño (AUNAR) (2025) ofreció el programa de Ingeniería Informática en modalidad a distancia, con un fuerte enfoque en desarrollo de software y procesamiento de grandes volúmenes de datos. Este programa contó con líneas de investigación activas en Desarrollo de Software y Sistemas Inteligentes, respaldadas por el grupo de investigación SEDMATEC, que demostró avances significativos en la medición de estándares reconocidos por Colciencias [38]. La existencia de estas líneas de investigación aportó un entorno propicio para el desarrollo de modelos innovadores, como la aplicación de técnicas de IA generativa y agentes inteligentes para la mejora del análisis de requisitos, fomentando así la consolidación de una base investigativa local comprometida con el avance tecnológico.

H. SUPUESTOS TEÓRICOS

Inteligencia Artificial y Tecnologías Relacionadas

Inteligencia Artificial (IA)

La IA fue una disciplina científica que buscó replicar procesos cognitivos humanos como el razonamiento, aprendizaje y toma de decisiones, utilizando modelos computacionales avanzados [39]. En esta investigación, se partió de que la IA podía aprender patrones relacionados con la calidad de los requisitos de software, detectando errores, inconsistencias o ambigüedades que podían afectar el ciclo de vida del desarrollo.

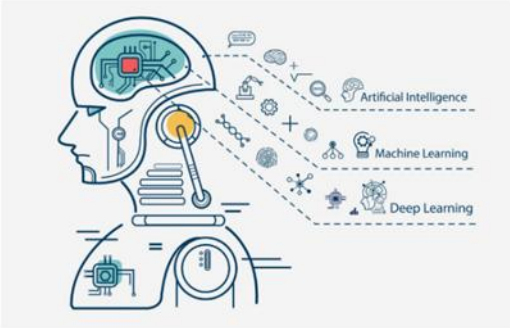


Fig 1. Inteligencia artificial (IA)

Nota: Fuente [40]

Aprendizaje Automático y Aprendizaje Profundo

El aprendizaje automático (ML) permitió que los sistemas aprendieran a partir de datos sin intervención humana directa. El aprendizaje profundo (DL), a través de redes neuronales jerárquicas, identificó patrones complejos y abstractos [41]. Se asumió que estos modelos, entrenados con conjuntos de requisitos reales, podían evaluar su calidad con un alto grado de precisión.

IA Generativa

La IA generativa, mediante modelos como GPT o BERT, tuvo la capacidad de generar lenguaje natural coherente, útil para mejorar, reescribir o validar requisitos de software. Este estudio propuso que la IA generativa podía asistir activamente en la redacción y evaluación semántica de requisitos, proponiendo mejoras basadas en contextos previos [22], [42].

Agentes Inteligentes

Un agente inteligente fue concebido como un sistema autónomo que percibe su entorno, toma decisiones y actúa para alcanzar un objetivo. Se consideró que un enfoque multiagente especializado podía dividir tareas complejas en componentes manejables, aumentando la eficiencia del análisis automatizado de requisitos [43].

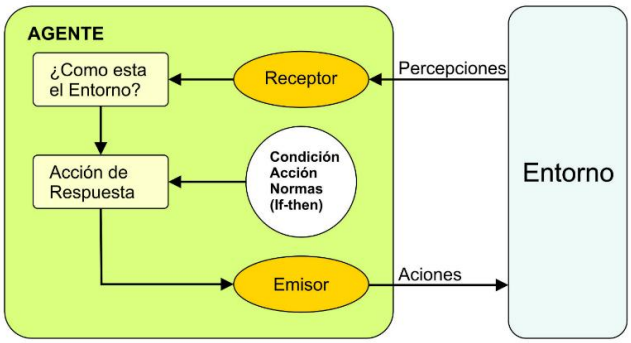


Fig 2. Agentes inteligentes

Nota: Fuente [44]

Integración de Agentes Inteligentes con Input Engineering (Ingeniería de Prompts)

La ingeniería de Prompt, o Input Engineering, optimizó la interacción con modelos generativos mediante el diseño preciso de entradas lingüísticas. Este estudio planteó que la combinación de agentes inteligentes con técnicas de Prompt Engineering mejoraba la calidad de las respuestas generadas por la IA [45]. Cada agente podía construir entradas optimizadas basadas en el contexto y en su tarea específica, incrementando la eficacia del análisis y detección de problemas en los requisitos.

Procesamiento de Lenguaje Natural (PLN)

El PLN permitió que las máquinas interpretaran y analizaran el lenguaje humano. Este proyecto asumió que, con PLN, era posible detectar ambigüedades, defectos gramaticales y contradicciones

en los requisitos, facilitando una evaluación lingüística exhaustiva a nivel sintáctico y semántico [46].

Fig 3. Procesamiento de lenguaje natural

Nota: Fuente [47]

Evaluación Automatizada de Calidad

Se consideró que, mediante métricas objetivas, modelos predictivos y análisis lingüístico, era posible evaluar automáticamente la calidad de los requisitos. Se sostuvo que los enfoques automáticos, asistidos por IA generativa y agentes inteligentes, eran más eficientes, escalables y exactos que los métodos tradicionales [48].

Integración Tecnológica: Frameworks y Herramientas

La investigación asumió que tecnologías como Python, frameworks web (Flask/Django), bibliotecas de aprendizaje automático (Scikit-learn, TensorFlow, Transformers) y entornos colaborativos (Jupyter, Streamlit) podían integrarse para construir un sistema modular y funcional, complementado con interfaces visuales para facilitar la interacción entre usuarios humanos y el sistema inteligente [49].

Ingeniería de Requisitos

Requisitos Funcionales

Los requisitos funcionales describieron las acciones específicas que un sistema debía ser capaz de realizar para satisfacer las necesidades del usuario. Estaban directamente ligados a las funcionalidades del sistema, como autenticación, generación de reportes o procesamiento de datos [23]. En esta investigación, se partió del supuesto de que los requisitos funcionales podían ser

identificados, validados y mejorados automáticamente mediante herramientas de IA, garantizando su correcta alineación con los objetivos del sistema

Fig 4. Requisitos funcionales

Nota: Fuente [50]

Calidad de Requisitos de Software

La calidad se fundamentó en atributos como claridad, completitud, consistencia, viabilidad, verificabilidad y rastreabilidad [51]. Se esperó que esta calidad podía ser evaluada de forma automática y objetiva mediante el uso de IA, reduciendo la subjetividad y los errores comunes en validaciones manuales.

Calidad de Requisitos Funcionales

La calidad de los requisitos funcionales se evaluó mediante atributos como claridad, precisión, no ambigüedad, consistencia, completitud y trazabilidad. Se supuso que estos atributos podían ser analizados por modelos computacionales que utilizaron PLN e IA generativa [52]. Esta evaluación automatizada permitió reducir errores humanos, acortar los ciclos de validación y asegurar requisitos funcionales bien definidos desde etapas tempranas del desarrollo.

Validación Semántica de Requisitos Funcionales

Se partió del supuesto de que, al integrar técnicas de PLN y modelos de lenguaje como GPT, era posible realizar una validación semántica de los requisitos funcionales, verificando su coherencia, intención y concordancia con el problema [53]. Esto proporcionaría un control de calidad más riguroso, ayudando a evitar interpretaciones erróneas que comprometieran la funcionalidad del sistema.

Análisis de Requisitos en Ingeniería de Software

Desde la ingeniería de software, los requisitos fueron elementos críticos que determinaron el diseño y funcionalidad del sistema. Este estudio partió del supuesto de que, en entornos ágiles, se requirió una validación rápida, automatizada y fiable para asegurar que los requisitos evolucionaban sin perder coherencia ni precisión [54].

Metodologías Ágiles

Las metodologías ágiles priorizaron la entrega continua de software funcional, la colaboración con el cliente y la adaptación al cambio [55]. En esta investigación se asumió que, debido al entorno dinámico del desarrollo de software, el uso de enfoques ágiles facilitó la iteración frecuente, la retroalimentación constante y la validación temprana de requisitos funcionales mediante herramientas automatizadas de IA [45].

Programación Extrema (XP)

XP fue una metodología ágil que promovió prácticas como desarrollo iterativo, pruebas continuas, retroalimentación rápida y programación en pareja. Este estudio partió del supuesto de que, al aplicar XP, los requisitos funcionales evolucionaron en ciclos cortos y podían ser validados constantemente por agentes inteligentes e IA generativa [56]. Se consideró que XP, al enfatizar la comunicación continua y el diseño evolutivo, se complementó eficazmente con la validación automatizada, mejorando la calidad desde las primeras fases del desarrollo.

I. VARIABLES DEL ESTUDIO

3) Definición nominal de variables

Variable Independiente (VI):

Modelo basado en agentes inteligentes con inteligencia artificial generativa.

Se refirió al sistema desarrollado que integró múltiples agentes inteligentes apoyados en modelos de lenguaje de gran escala (LLMs), con el propósito de analizar de manera automatizada los requisitos de software, identificar errores y proponer mejoras en su redacción y estructura.

Variable Dependiente (VD):

Calidad de los requisitos de software.

Hizo referencia al nivel de cumplimiento de los atributos de calidad en los requisitos funcionales, tales como claridad, completitud, consistencia, verificabilidad y ausencia de ambigüedad. Esta variable también consideró la reducción de errores en los requisitos y el nivel de comprensión que estos generaron en los usuarios o evaluadores.

4) Definición operativa de variables

Variable Independiente (VI):

- Modelo basado en agentes inteligentes con inteligencia artificial generativa.
- Indicador: Uso del modelo en el proceso de validación.
- Medición: Aplicación del modelo en la herramienta desarrollada para analizar requisitos.
- Escala: Cualitativa dicotómica (Uso del modelo / No uso del modelo).
- Técnica: Ejecución de pruebas en dos escenarios: validación manual y validación con herramienta.

Variable Dependiente (VD):

Calidad de los requisitos de software.

Se evaluó a través de los siguientes indicadores:

- Reducción de errores en los requisitos
- Indicador: Número de errores detectados.
- Medición: Cantidad de errores identificados en cada requisito.
- Escala: Cuantitativa (número de errores).
- Ambigüedad en los requisitos
- Indicador: Presencia de ambigüedades.
- Medición: Identificación de términos ambiguos o poco claros.
- Escala: Cuantitativa (frecuencia de ambigüedades detectadas).
- Comprensión de los requisitos
- Indicador: Nivel de entendimiento por parte de los evaluadores.
- Medición: Evaluación mediante formulario de percepción aplicado a los usuarios.
- Escala: Cualitativa ordinal (bajo, medio, alto).
- Tiempo de validación

- Indicador: Tiempo empleado en el análisis.
- Medición: Minutos utilizados para validar requisitos.
- Escala: Cuantitativa (minutos).

J. FORMULACIÓN DE HIPÓTESIS

1) Hipótesis de investigación

El uso de un modelo basado en IA generativa y agentes inteligentes mejoró significativamente la calidad de los requisitos en proyectos de desarrollo de software, redujo errores y disminuyó el tiempo de validación.

2) Hipótesis nula

El uso de un modelo basado en IA generativa y agentes inteligentes no generó mejoras significativas en la calidad de los requisitos ni redujo el tiempo de validación en los proyectos de software.

3) Hipótesis alterna

Existió una diferencia significativa positiva en la calidad de los requisitos y en el tiempo de validación entre proyectos que utilizaron herramientas tradicionales y aquellos que implementaron el modelo propuesto basado en IA generativa y agentes inteligentes.

III. METODOLOGÍA

A. PARADIGMA

Esta investigación tuvo un enfoque positivista debido a que su principal objetivo fue analizar y evaluar los resultados del modelo desarrollado, además se orientó hacia la obtención de resultados objetivos, medibles y verificables mediante el uso de técnicas que facilitaron esta tarea. Todo esto se implementó en un modelo computacional con el cual se buscó comprobar y mejorar el análisis de calidad de requisitos de software, empleando diversas herramientas como IA generativa que trabajaron a su vez con agentes inteligentes para obtener resultados satisfactorios que también fueron avalados y probados por expertos en elicitación de requisitos. Esto ayudó a comprobar si los resultados obtenidos por el modelo fueron reales y medibles mediante métricas de calidad.

B. ENFOQUE

El enfoque de esta investigación fue cuantitativo y aplicado, porque se centró en el desarrollo y la validación del modelo computacional mediante datos medibles y análisis empírico. El objetivo fue desarrollar una solución tecnológica que permitiera evaluar la calidad de requisitos de software mediante técnicas de IA generativa, incluyendo agentes inteligentes. Este enfoque permitió la recopilación de datos objetivos mediante pruebas controladas, lo que facilitó el establecimiento de relaciones entre el uso y las mejoras en el análisis de requisitos del modelo. Además, se buscó que el modelo fuera utilizado tanto en la academia, como en el campo laboral y de investigación.

C. MÉTODO

El método utilizado en esta investigación fue experimental y tuvo como objetivo verificar la efectividad del modelo propuesto mediante la observación controlada del desempeño en un contexto claro y definido. Se evaluó la capacidad de probar y aprobar requisitos de software mediante la implementación de prototipos funcionales para identificar atributos de calidad como claridad, integridad y consistencia. Este método permitió probar la utilidad de un modelo y medir resultados bajo condiciones específicas, y también observar si realmente fue una ayuda para mejorar el análisis de calidad en los requisitos de software, para de esta manera realizar los ajustes necesarios.

D. TIPO DE INVESTIGACIÓN

El tipo de investigación fue aplicada y orientada a la tecnología, y tuvo como objetivo resolver un problema específico en el campo del desarrollo de software, específicamente el análisis de requisitos de calidad. Utilizando IA generativa y agentes inteligentes, se desarrollaron modelos funcionales que contribuyeron a mejorar el proceso de validación de requisitos en proyectos reales. Esta investigación aplicada se centró en la aplicación práctica del conocimiento, mientras que la dimensión técnica implicó el diseño, desarrollo y evaluación de equipos de computación que pudieron usarse tanto en contextos productivos como académicos, y buscó no solo estudiar el modelo sino también crear una herramienta útil.

K. DISEÑO DE LA INVESTIGACIÓN

El diseño de esta investigación fue de tipo cuasiexperimental con pruebas piloto, ya que se buscó evaluar cómo el uso del modelo propuesto influyó en la calidad de los requisitos de software. Para ello, se compararon dos formas de trabajo: la validación manual tradicional y la validación utilizando la herramienta desarrollada.

Se consideró cuasiexperimental porque se introdujo un cambio concreto en el proceso, el uso del modelo basado en agentes inteligentes con IA generativa, y se analizaron sus efectos en aspectos como el tiempo de validación y la cantidad de errores detectados. Esto permitió observar de manera directa si la herramienta realmente generaba mejoras frente al método tradicional.

Además, la validación se realizó mediante pruebas piloto con usuarios, específicamente estudiantes que simularon equipos de trabajo reales. Aunque no se trató de un experimento completamente controlado como en laboratorio, sí se mantuvieron condiciones similares para todos los grupos, lo que permitió hacer comparaciones válidas entre los resultados obtenidos.

El uso de pruebas piloto fue clave, ya que permitió evaluar el modelo en un entorno cercano a la realidad, identificar posibles fallas y obtener una primera retroalimentación sobre su funcionamiento. De esta manera, no solo se evaluó el modelo desde lo técnico, sino también desde su utilidad en un contexto real de uso.

TABLA I. Metodología Preliminar

FASE	ACTIVIDAD	RESULTADOS	RECURSOS

Caracterización de requisitos	Caracterizar los atributos esenciales de un requisito de software bien definido.	Atributos de un requisito bien definido para la implementación del modelo.	Investigación en ingeniería de requisitos, herramientas de análisis de texto.
	Identificar las características clave que afectan la calidad del requisito.	Listado de características clave que determinan la calidad de un requisito.	Investigación en calidad de requisitos, análisis comparativo.
Desarrollo del modelo	Definir el modelo automatizado de análisis de requisitos mediante IA generativa.	Modelo estructurado para la evaluación automática de los requisitos.	LLMs (GPT-4 o similares), herramientas de Input Engineering.
	Configurar la comunicación entre agentes inteligentes para el análisis.	Comunicación fluida entre los agentes para detectar errores en los requisitos.	Python, LLMs, Input Engineering,
Desarrollo de la herramienta	Desarrollar una herramienta que integre los agentes inteligentes para procesar y analizar requisitos.	Herramienta de software funcional que asiste a los desarrolladores en el análisis de requisitos.	Python (Flask o Django), LLMs, Input Engineering.
	Integrar la funcionalidad de análisis de inconsistencias y errores en la herramienta.	Herramienta que detecta automáticamente inconsistencias y errores en los requisitos.	Python, Input Engineering, LLMs.

Validación y evaluación	Aplicar un método de validación con usuarios para evaluar la efectividad de la herramienta.	Resultados de la evaluación de la herramienta con usuarios.	Encuestas, métricas de validación, usuarios (desarrolladores).
	Medir el impacto en la precisión de los requisitos y la reducción de errores.	Análisis de precisión y reducción de errores en requisitos mediante la herramienta.	Herramientas de análisis estadístico (Python, Matplotlib, Seaborn).
	Comparar el desempeño de la herramienta con procesos manuales o tradicionales.	Comparación de desempeño entre la herramienta automatizada y métodos manuales.	Herramientas estadísticas, datos de validación.

L. POBLACIÓN

La población de estudio estuvo conformada por dos componentes principales. En primer lugar, se consideraron documentos de requisitos de software provenientes de proyectos reales o simulados, utilizados como base para el análisis automatizado. Estos documentos incluyeron descripciones funcionales y no funcionales de sistemas desarrollados en contextos académicos, comerciales o de código abierto.

En segundo lugar, la población incluyó usuarios en formación, específicamente estudiantes de Ingeniería de Sistemas, quienes representaron un grupo relevante para la evaluación del modelo en un contexto de uso realista. Estos participantes contaron con conocimientos básicos en ingeniería de requisitos, lo que permitió analizar la utilidad del sistema en escenarios académicos similares a entornos profesionales.

En conjunto, la población correspondió tanto al amplio conjunto de requisitos de software que podían ser analizados mediante el modelo propuesto, como a los usuarios potenciales que interactuaron con este tipo de herramientas en procesos de validación de requisitos.

M. MUESTRA

Muestra de investigación (documentos)

En este proyecto se trabajó con una muestra intencional de entre 10 y 15 documentos de requisitos de software, los cuales contaban con características adecuadas para su evaluación, como la presencia de errores comunes (ambigüedad, incompletitud e inconsistencia) y una estructura definida. Estos documentos fueron seleccionados de manera no probabilística, ya que permitieron poner a prueba el funcionamiento del modelo en escenarios representativos, sin necesidad de utilizar un gran volumen de información, dado que el objetivo principal fue validar el comportamiento del sistema más que realizar un análisis estadístico.

Muestra de aplicación (validación con usuarios)

Para la validación del modelo, se contó con la participación de 30 estudiantes de quinto semestre del programa de Ingeniería de Sistemas, quienes se encontraban cursando la asignatura de Ingeniería de Requisitos. Los participantes se organizaron en 6 grupos de 5 integrantes, simulando equipos de trabajo reales dentro de un entorno académico.

El perfil de los participantes fue adecuado para la prueba, ya que contaban con conocimientos básicos en análisis y especificación de requisitos, lo que permitió obtener resultados coherentes y comparables durante la actividad.

Durante la validación, se trabajó con dos casos de estudio:

- FoodExpress, una aplicación orientada a la gestión de pedidos de comida.
- SaludYa, un sistema enfocado en la gestión de citas médicas.

Estos casos fueron seleccionados por ser escenarios comprensibles y cercanos a la realidad, lo que facilitó la elaboración y el análisis de requisitos por parte de los participantes.

La muestra, aunque pequeña, se consideró representativa para los fines del estudio, ya que permitió evaluar el comportamiento del modelo en un contexto controlado y con usuarios reales. Además, al trabajar con grupos organizados y condiciones similares, se garantizó la consistencia en los resultados obtenidos, lo que permitió comparar de manera adecuada el método manual frente al uso de la herramienta.

N. TÉCNICAS DE RECOLECCIÓN DE INFORMACIÓN

Esta investigación empleó dos métodos principales para la obtención de datos: el análisis documental y el registro automatizado de los resultados del sistema. El análisis documental facilitó el estudio de los documentos de requisitos de software utilizados como insumos fundamentales para el modelo, lo que permitió evaluar su composición, articulación y atributos técnicos. Al mismo tiempo, el modelo implementado de forma autónoma generó datos sobre la calidad de los requisitos previamente evaluados, los cuales se recopilaron mediante registros de resultados y métricas específicas de rendimiento.

Validez de la técnica

Para asegurar el correcto funcionamiento de las técnicas utilizadas para la recolección de información, se emplearon documentos de requisitos que siguieron estándares reconocidos, como la norma ISO/IEC/IEEE 29148, la cual estableció directrices para la especificación y estructura de requisitos. Los resultados obtenidos, tras ser analizados por el modelo, fueron revisados y validados por expertos en ingeniería de software, con el fin de verificar que el sistema realmente midiera lo que se esperaba.

Confiabilidad de la técnica

Para garantizar que la información recolectada fuera confiable, es decir, que los resultados fueran estables y reproducibles, se realizaron al menos tres repeticiones del análisis con el mismo modelo, utilizando tanto el mismo conjunto de documentos como otros diferentes. Si el modelo generaba respuestas similares en situaciones comparables, se podía confiar en su adecuado funcionamiento. Además, se verificó que el sistema no modificara sus resultados sin justificación y que analizara los documentos de manera consistente. De este modo, se aseguró que la herramienta fuera estable y no dependiera del momento o de quien la utilizara.

O. INSTRUMENTO DE RECOLECCIÓN DE DATOS

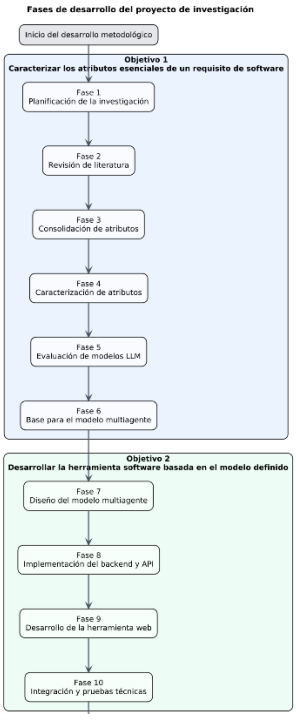
En este proyecto se utilizaron dos instrumentos principales para la recolección de información. El primero estuvo conformado por los documentos de requisitos de software, específicamente casos de uso e historias de usuario, los cuales describieron las funcionalidades que debía cumplir un sistema. Estos documentos constituyeron la base de trabajo del modelo. El segundo instrumento correspondió al sistema desarrollado, el cual analizó dichos documentos y generó resultados automáticos, como listas de ambigüedades y puntuaciones de completitud y claridad, entre otros

indicadores. Toda esta información se almacenó y revisó con el fin de verificar si el modelo realmente funcionaba conforme a lo esperado.

P. DESARROLLO METODOLÓGICO

Para el cumplimiento de los objetivos planteados en la investigación, se estructuró un desarrollo metodológico organizado en un conjunto de fases secuenciales que abarcaron desde la identificación y el análisis de los atributos de calidad de los requisitos hasta el desarrollo, la implementación y la posterior validación de la solución tecnológica.

Este proceso permitió articular de manera coherente las actividades correspondientes a cada objetivo específico, garantizando un flujo de trabajo ordenado y orientado a la obtención de resultados. La Figura 6 presentó, de forma general, las fases que compusieron el desarrollo metodológico del proyecto.



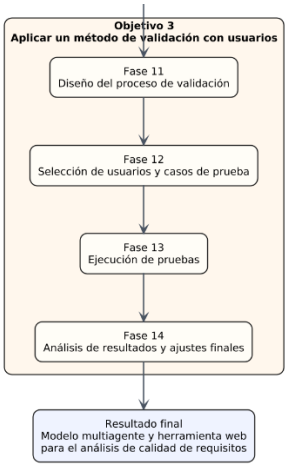


Fig. 5. Fases del desarrollo metodológico.

a) Fase 1. Planificación de la investigación

La primera fase del desarrollo metodológico correspondió a la planificación de la investigación, en la cual se establecieron las bases que orientaron todo el proceso. En esta etapa se definió el enfoque general del estudio, así como los criterios que guiaron la búsqueda, la selección y el análisis de la información.

Asimismo, se determinaron las fuentes por consultar, considerando tanto artículos científicos como normativas reconocidas en el ámbito de la ingeniería de requisitos, con el objetivo de garantizar la calidad y la pertinencia de la información. En total, se proyectó la revisión de 24 documentos relevantes, los cuales sirvieron como soporte teórico para el desarrollo de la investigación.

Adicionalmente, se definieron los lineamientos metodológicos que permitieron organizar y estructurar las fases posteriores, asegurando la coherencia del proceso y una adecuada articulación entre los objetivos planteados y las actividades desarrolladas.

b) Fase 2. Revisión de literatura

Una vez definida la planificación de la investigación, se llevó a cabo la fase de revisión de literatura, en la cual se analizaron en detalle las 24 fuentes previamente seleccionadas, entre artículos científicos y normativas relacionadas con la calidad de los requisitos de software.

Durante esta etapa, se realizó una lectura crítica de cada documento con el propósito de identificar los atributos, criterios y enfoques utilizados en la evaluación de los requisitos dentro de la ingeniería de software. Se prestó especial atención a aquellos elementos que permitieron

comprender cómo diferentes autores abordaron aspectos como la claridad, completitud, consistencia y verificabilidad de los requisitos.

Como resultado de esta fase, se obtuvo un conjunto amplio de atributos de calidad identificados a partir de las diferentes fuentes analizadas. Sin embargo, estos atributos se encontraban inicialmente dispersos, con diferentes denominaciones y enfoques según cada autor, lo que generó la necesidad de realizar un proceso de organización y depuración en la siguiente fase del desarrollo metodológico.

c) Fase 3. Consolidación de atributos

Posteriormente, se desarrolló la fase de consolidación de atributos, en la cual se llevó a cabo un análisis detallado de los atributos identificados durante la revisión de literatura. En esta etapa, se buscó comprender las diferentes perspectivas propuestas por los autores, identificando similitudes conceptuales, diferencias terminológicas y posibles redundancias en la manera en que se describieron los atributos de calidad de los requisitos de software.

Para ello, se realizó un proceso sistemático de comparación entre los atributos identificados, analizando su significado, contexto de uso y relación con otros atributos. Este análisis permitió evidenciar que, aunque muchos autores emplearon términos distintos, en numerosos casos hicieron referencia a características equivalentes o complementarias.

A partir de lo anterior, se procedió a agrupar aquellos atributos que presentaban semejanzas conceptuales o semánticas, con el objetivo de reducir la dispersión de la información y establecer una base más coherente para el estudio. Este proceso implicó la depuración de términos redundantes y la unificación de conceptos, facilitando una mejor comprensión de los elementos que influyen en la calidad de los requisitos.

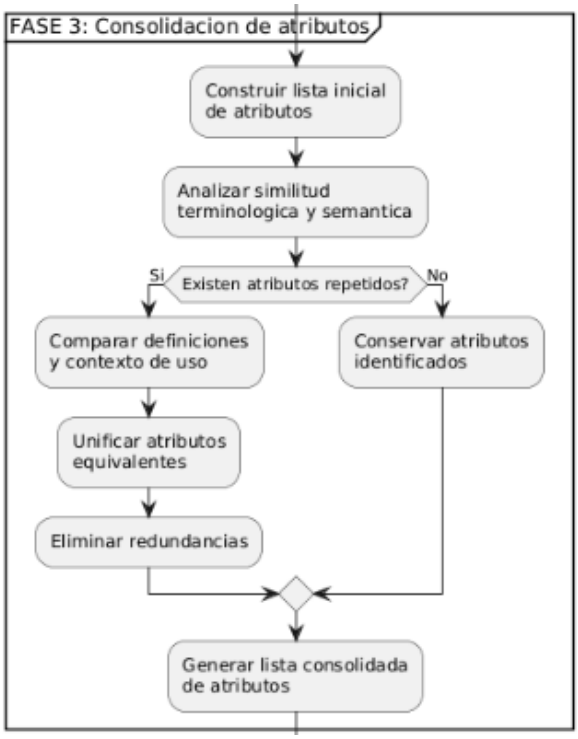


Fig. 6. Proceso de consolidación de atributos.

La Figura 6 presentó el proceso seguido durante esta fase, en el cual se ilustraron las actividades de comparación, agrupación y depuración de los atributos identificados, evidenciando la transición desde un conjunto inicial disperso hacia una estructura más organizada.

Es importante señalar que los resultados obtenidos a partir de esta fase, particularmente la consolidación final de los atributos, se presentaron en el Capítulo IV, correspondiente a los resultados de la investigación, específicamente en la tabla en la que se detallaron y organizaron los atributos consolidados derivados de este proceso.

d) Fase 4. Caracterización de atributos

En esta etapa, se llevó a cabo la fase de caracterización de atributos, en la cual se profundizó en el estudio de los atributos previamente consolidados, con el propósito de definir de manera precisa sus características y facilitar su aplicación en el análisis de requisitos de software.

En esta fase, cada atributo fue analizado individualmente, estableciendo su significado dentro del contexto de la calidad de los requisitos, así como su alcance y forma de interpretación. De igual manera, se incorporaron elementos descriptivos que permitieron comprender mejor su uso,

incluyendo ejemplos que ilustraron tanto su correcta formulación como posibles errores en su aplicación.

Este trabajo permitió organizar los atributos desde una perspectiva más detallada, pasando de una visión general a una definición más específica y funcional, lo que favoreció su posterior incorporación en el modelo multiagente propuesto en la investigación.

Los resultados obtenidos en esta fase, particularmente la matriz de caracterización de los atributos de calidad, se presentaron en el Capítulo IV, correspondiente a los resultados de la investigación, específicamente en la tabla donde se describieron los atributos definidos junto con sus principales características.

e) Fase 5. Evaluación de modelos LLM

En esta etapa, se llevó a cabo la evaluación de modelos de lenguaje de gran escala (LLM), con el propósito de seleccionar la herramienta más adecuada para apoyar el análisis automatizado de la calidad de los requisitos de software.

Para ello, se realizó un análisis comparativo entre diferentes modelos de inteligencia artificial generativa, considerando criterios relevantes como la capacidad de comprensión del lenguaje natural, la coherencia en las respuestas, el razonamiento lógico, la adaptabilidad al contexto y la facilidad de integración en un entorno de desarrollo.

Asimismo, se revisaron estudios y documentos que presentaron evaluaciones previas de estos modelos en distintos contextos, especialmente en tareas relacionadas con la ingeniería de software, con el fin de contar con un sustento teórico para la selección del modelo más pertinente.

Este análisis permitió contrastar las características y capacidades de los modelos evaluados, facilitando la toma de decisiones respecto del modelo que mejor se ajustó a los requerimientos de la investigación.

Es importante señalar que los resultados obtenidos en esta fase, particularmente la comparación de los modelos y la selección del modelo más adecuado, se presentaron en el Capítulo IV, correspondiente a los resultados de la investigación, donde se detalló el análisis realizado y los criterios que sustentaron dicha elección.

f) Fase 6. Base para el modelo multiagente

En esta etapa, se definieron los fundamentos para la construcción del modelo multiagente propuesto en la investigación, integrando los elementos obtenidos en las fases anteriores.

Para ello, se tomaron como referencia los atributos de calidad de los requisitos previamente caracterizados, los cuales fueron establecidos como criterios base para el análisis dentro del sistema. Asimismo, se consideró el modelo de lenguaje seleccionado, con el fin de definir su papel dentro del procesamiento de información en lenguaje natural.

De igual manera, se planteó la estructura conceptual del modelo multiagente, identificando los roles que asumieron los agentes y las funciones asociadas a cada uno dentro del análisis de los requisitos de software. Esto permitió definir un enfoque organizado en el que cada agente abordó diferentes aspectos de la calidad de los requisitos.

Adicionalmente, se establecieron los lineamientos para el diseño de los Prompt que orientaron el comportamiento de los agentes, permitiendo dirigir el análisis hacia los atributos de calidad definidos en la investigación.

Esta fase permitió consolidar las bases conceptuales necesarias para el desarrollo del modelo multiagente. A partir de estos fundamentos, en el Capítulo IV se presentó la implementación del modelo propuesto, junto con su representación estructural y su funcionamiento.

g) Fase 7. Modelo multiagente

A partir de la arquitectura presentada, se observó que el modelo organizó el proceso de análisis en diferentes componentes que permitieron gestionar la entrada de los requisitos, su procesamiento y la generación de resultados.

En este contexto, cada agente cumplió una función específica dentro del sistema, abordando distintos aspectos de la calidad de los requisitos. Esta distribución permitió realizar un análisis más completo, ya que combinó diferentes perspectivas dentro de un mismo flujo de evaluación.

Adicionalmente, el modelo incorporó el uso de Prompt que orientaron el comportamiento de los agentes, lo que permitió que el modelo de lenguaje interpretara los requisitos y generara respuestas alineadas con los criterios definidos en la investigación.

En conjunto, el modelo multiagente propuesto permitió estructurar el análisis de los requisitos de software de manera organizada, facilitando la identificación de problemas y la generación de mejoras en los requisitos evaluados.

h) Fase 8. Implementación del backend y API del modelo

Se desarrolló la implementación del Backend del sistema, el cual permitió gestionar el procesamiento de los requisitos de software y la interacción con el modelo multiagente propuesto.

El Backend se encargó de recibir los requisitos ingresados por el usuario, procesarlos a través de la lógica definida en el modelo y coordinar la ejecución de los agentes responsables del análisis. De esta manera, se estableció un flujo de procesamiento que permitió evaluar los requisitos considerando los atributos de calidad definidos en la investigación.

Adicionalmente, se implementó una API que facilitó la comunicación entre el Backend y otros componentes del sistema, permitiendo el envío y la recepción de información de manera estructurada. Esta API actuó como un punto de acceso para el modelo, posibilitando su integración con la herramienta web y con otros posibles sistemas externos.

A través de esta arquitectura, el modelo multiagente pudo ser consumido como un servicio, lo que permitió ejecutar el análisis de requisitos de forma flexible y escalable. Asimismo, la separación entre el Backend y la API contribuyó a una mejor organización del sistema y facilitó su mantenimiento y extensión.

En conjunto, esta implementación permitió materializar el modelo propuesto, llevando su funcionamiento a un entorno práctico en el que fue posible procesar requisitos de software y obtener resultados de análisis de manera automatizada.

i) Fase 9. Desarrollo de la herramienta web

En esta etapa, se llevó a cabo el desarrollo de la herramienta web orientada a la gestión de proyectos, la cual fue concebida como el entorno de interacción para la aplicación del modelo multiagente propuesto.

Para su implementación, se utilizaron tecnologías modernas de desarrollo web, empleando Angular para la construcción del Frontend, Node.js con Express para el desarrollo del Backend y PostgreSQL como sistema de gestión de bases de datos. Esta arquitectura permitió estructurar una solución organizada y escalable para la gestión de la información.

La herramienta fue diseñada para permitir el registro y la administración de proyectos, así como la creación y gestión de requisitos de software. Asimismo, se definió la integración con la API del modelo multiagente, mediante la cual los requisitos podían ser enviados para su análisis automático.

Durante esta fase, se establecieron los componentes principales de la aplicación, incluyendo la interfaz de usuario, la lógica de negocio y la estructura de almacenamiento de datos, con el fin de garantizar una correcta interacción entre el usuario y el sistema.

Esta etapa permitió definir la base funcional de la herramienta web, la cual fue utilizada posteriormente en el proceso de validación del modelo. Es importante señalar que los resultados asociados a la implementación de la herramienta, así como su funcionamiento e interfaz, se presentaron en el Capítulo IV, correspondiente a los resultados de la investigación.

j) Fase 10. Integración y pruebas técnicas

En esta etapa, se llevó a cabo la integración de los diferentes componentes del sistema, con el objetivo de garantizar el correcto funcionamiento de la herramienta web en conjunto con el modelo multiagente y la API desarrollada.

Durante este proceso, se estableció la comunicación entre el Frontend, el Backend y el modelo, verificando que el flujo de información se realizara de manera adecuada desde el ingreso de los requisitos hasta la generación de los resultados del análisis.

Posteriormente, se realizaron pruebas técnicas orientadas a validar el comportamiento del sistema, incluyendo pruebas funcionales sobre los módulos de gestión de proyectos, registro de requisitos y ejecución del análisis automatizado. Estas pruebas permitieron comprobar la correcta interacción entre los componentes y el cumplimiento de las funcionalidades definidas.

Adicionalmente, se evaluó el flujo completo del sistema, verificando que los requisitos ingresados fueran procesados correctamente por el modelo multiagente y que los resultados generados fueran coherentes con los criterios de calidad establecidos.

Finalmente, se llevó a cabo el despliegue de la aplicación en un entorno de hosting, lo que permitió validar su funcionamiento en un entorno real y asegurar su disponibilidad para el acceso de los usuarios.

Esta etapa permitió confirmar la correcta integración de la solución propuesta, dejando el sistema preparado para su posterior validación con usuarios en las siguientes fases del proyecto.

k) Fase 11. Diseño del proceso de validación

En la Fase 11, se diseñó el proceso de validación del modelo propuesto, definiendo la dinámica experimental, las variables de estudio y las condiciones bajo las cuales se llevaría a cabo la evaluación. Esta fase tuvo como propósito establecer un entorno controlado que permitiera comparar el desempeño del método tradicional de validación de requisitos frente al uso de la herramienta basada en agentes inteligentes con IA generativa.

Dentro del diseño metodológico, se definieron como variables dependientes el tiempo de análisis y la cantidad de errores detectados en los requisitos de software, mientras que la variable independiente correspondió al uso del modelo propuesto integrado en la herramienta desarrollada.

La investigación se clasificó como de tipo cuasiexperimental debido a que se introdujo una intervención específica dentro del proceso de validación —el uso de la herramienta automatizada— y, posteriormente, se compararon sus efectos frente a un escenario tradicional de análisis manual. Aunque no existió una asignación aleatoria completa de los participantes ni un entorno totalmente controlado, como en un laboratorio, sí se establecieron condiciones homogéneas para todos los grupos participantes, lo que permitió realizar comparaciones válidas entre ambos métodos de trabajo.

Asimismo, el estudio presentó características experimentales debido a que:

- Se manipuló una variable independiente concreta (uso de la herramienta basada en IA generativa).
- Se observaron sus efectos sobre variables medibles.

- Se mantuvieron condiciones similares para todos los participantes.
- Se aplicaron procedimientos estandarizados durante las pruebas.
- Se realizaron comparaciones directas entre dos escenarios de validación.

De esta manera, el diseño permitió evaluar objetivamente el impacto del modelo propuesto sobre el proceso de validación de requisitos, garantizando coherencia metodológica con el enfoque cuantitativo y experimental planteado en la investigación.

l) Fase 12. Diseño del proceso de validación

Posteriormente, en la Fase 12, se realizó la selección de los participantes y de los casos de prueba utilizados durante la validación del modelo.

La muestra estuvo conformada por 30 estudiantes de quinto semestre del programa de Ingeniería de Sistemas de la Universidad CESMAG, pertenecientes a la asignatura de Ingeniería de Requisitos. Los participantes fueron seleccionados debido a que contaban con conocimientos básicos sobre análisis y especificación de requisitos de software, lo que permitió desarrollar la actividad en condiciones similares a un entorno real de trabajo.

Con el propósito de simular dinámicas colaborativas propias de equipos de desarrollo de software, los participantes se organizaron en seis grupos de cinco integrantes. Cada grupo trabajó bajo las mismas condiciones metodológicas, utilizando los mismos tiempos de trabajo, instrucciones y formatos de evaluación, con el fin de reducir variaciones externas que pudieran afectar los resultados.

Asimismo, se definieron dos casos de estudio utilizados como base para la construcción y validación de requisitos funcionales:

- FoodExpress, una aplicación orientada a la gestión de pedidos de comida.
- SaludYa, un sistema enfocado en la gestión de citas médicas.

Estos casos fueron seleccionados por representar escenarios comprensibles, cercanos a contextos reales y adecuados para la identificación de problemas comunes en la redacción de requisitos, tales como ambigüedad, inconsistencia, incompletitud y falta de verificabilidad.

La utilización de casos homogéneos para todos los grupos permitió mantener condiciones controladas durante el proceso de validación, fortaleciendo la confiabilidad de las comparaciones realizadas entre el método manual y el uso de la herramienta desarrollada.

m) Fase 13. Ejecución de pruebas

En la Fase 13, se llevó a cabo la ejecución de las pruebas de validación bajo un esquema comparativo entre dos escenarios de análisis: la validación manual tradicional y la validación asistida mediante la herramienta basada en agentes inteligentes con IA generativa.

Inicialmente, cada grupo elaboró un conjunto de requisitos funcionales a partir de los casos de estudio definidos previamente. Posteriormente, los requisitos fueron intercambiados entre los grupos con el fin de evitar sesgos derivados del conocimiento previo sobre los documentos creados.

La dinámica experimental se desarrolló en dos etapas:

1. **Escenario de validación manual:**

Los participantes realizaron el análisis de los requisitos utilizando únicamente revisión humana tradicional, identificando errores relacionados con claridad, consistencia, ambigüedad, completitud y verificabilidad.

2. **Escenario de validación utilizando la herramienta:**

Posteriormente, los mismos requisitos fueron evaluados mediante la herramienta desarrollada, la cual integró el modelo multiagente y técnicas de IA generativa para detectar errores y generar observaciones automáticas.

Durante ambos escenarios se registraron métricas cuantitativas relacionadas con:

- Tiempo requerido para realizar el análisis.
- Cantidad de errores detectados.
- Diferencias entre ambos métodos de validación.

Adicionalmente, se aplicó un formulario de percepción orientado a evaluar la aceptación de la herramienta por parte de los participantes, considerando aspectos como facilidad de uso, utilidad percibida, apoyo al análisis y rapidez en el proceso de validación.

El desarrollo de esta fase permitió establecer una comparación directa entre ambos métodos bajo condiciones similares de trabajo, lo que fortaleció el carácter cuasiexperimental del estudio y permitió evaluar objetivamente el impacto del modelo propuesto en el análisis de calidad de requisitos de software.

n) Fase 14. Análisis de resultados y ajustes finales

Finalmente, en la Fase 14, se realizó la consolidación y el análisis de los datos obtenidos durante las pruebas de validación, permitiendo comparar el desempeño del método manual frente al uso de la herramienta basada en agentes inteligentes con IA generativa. Para ello, se analizaron variables como el tiempo empleado en el proceso de validación y la cantidad de errores detectados en los requisitos de software.

A partir de esta comparación, fue posible evaluar la efectividad del modelo propuesto en condiciones similares para todos los participantes, identificando diferencias en eficiencia y capacidad de detección de errores entre ambos escenarios de análisis. Asimismo, los resultados obtenidos permitieron validar el aporte de la herramienta dentro del proceso de ingeniería de requisitos y realizar ajustes finales orientados a mejorar su funcionamiento, usabilidad y precisión en el análisis automatizado de requisitos.

IV. RESULTADOS DE LA INVESTIGACIÓN

A. OBJETIVO ESPECÍFICO 1

Como resultado de la etapa de consolidación de atributos de calidad en los requisitos de software, se obtuvo un conjunto de atributos unificados a partir del análisis de los diversos documentos estudiados. Esta etapa facilitó la integración de diferentes conceptos presentes en la literatura, agrupándolos de acuerdo con sus similitudes conceptuales y evitando redundancias.

En la Tabla II se presentó la consolidación de atributos de calidad, en la cual se detallaron los atributos obtenidos, los términos unificados, sus respectivas fuentes y la justificación de su inclusión.

TABLA II. Consolidación de atributos de calidad identificados en la literatura.

Atributo (síntesis)	Qué se fusionó	Autores / Artículos en Word	Justificación de selección
1. Validez / Corrección	"Corrección", "Validez", "Correcto"	IEEE 830-1998; ISO/IEC/IEEE 29148:2018; Procedimiento validación requisitos; Ingeniería de Requisitos – Revisión Sistemática; Problem-Based SRS; A Case Study upon Non-Functional Requirements of Online Banking System; Requirements Engineering: A Roadmap (1999); Atomicity in Requirement Specification (2008); Requirements Validation Techniques Survey; Systematic Review of Requirements Engineering Practices	Se repite en muchas fuentes como que el requisito debe reflejar una necesidad real y válida de los stakeholders.
2. Claridad / No ambigüedad	"No ambiguo", "Claridad", "Comprensible", "Legibilidad"	IEEE 830-1998; ISO/IEC/IEEE 29148:2018; Metamorfosis; FRSL; Ambiguity Detection Techniques; Reducción de ambigüedades (OO models); Resolución de ambigüedades (NLP); Requirements Engineering in Software Houses; Systematization of Requirements;	Casi todos los artículos resaltan que el requisito debe ser claro, comprensible y sin interpretaciones múltiples.

		Problem-Based SRS; Requirements Engineering: A Roadmap (1999); Requirement Patterns for Business Modeling; Requirements Engineering Education Studies	
3. Completitud	"Completo"	IEEE 830-1998; ISO/IEC/IEEE 29148:2018; Estimación tamaño funcional; Procedimiento validación requisitos; Problem-Based SRS; Systematization of Requirements; A Case Study upon Non-Functional Requirements of Online Banking System; Requirements Validation Techniques Survey; Factors Affecting Requirements Engineering Process Success; Requirement Patterns for Business Modeling	Un requisito incompleto genera fallas. Se mantuvo porque es uno de los criterios más universales.
4. Consistencia	"Consistente"	IEEE 830-1998; ISO/IEC/IEEE 29148:2018; Metamorfosis; Ingeniería de Requisitos – Revisión Sistemática; Procedimiento validación requisitos; Systematization of Requirements; Implementation of Decision Support; Requirements Engineering Education Studies; Systematic Review of Requirements Engineering Practices	Todos coinciden en que los requisitos no deben contradecirse entre sí.

5. Viabilidad / Factibilidad	"Factibilidad", "Viabilidad", "Realizabilidad"	ISO/IEC/IEEE 29148:2018; Ingeniería de Requisitos – Revisión Sistemática; Metamorfosis; Critical Success Factors; Implementation of Decision Support; A Case Study upon Non-Functional Requirements of Online Banking System; Requirements Engineering: A Roadmap (1999); Factors Affecting Requirements Engineering Process Success; Requirement Patterns for Business Modeling; Systematic Review of Requirements Engineering Practices	Se agruparon términos porque todos refieren a que sea posible implementar el requisito con los recursos disponibles.
6. Priorización	"Clasificado por importancia", "Estabilidad"	IEEE 830-1998; ISO/IEC/IEEE 29148:2018; Estimación tamaño funcional; Implementation of Decision Support; Requirements Engineering in Software Houses; A Case Study upon Non-Functional Requirements of Online Banking System; Requirements Engineering: A Roadmap (1999); Factors Affecting Requirements Engineering Process Success	Se mantuvo porque permite planificar qué implementar primero según importancia.
7. Trazabilidad	"Rastreabilidad", "Trazable"	IEEE 830-1998; ISO/IEC/IEEE 29148:2018; Procedimiento validación requisitos; Problem-Based SRS; Systematization of Requirements; Reducción de ambigüedades (OO models); Requirements Engineering in Software Houses; Requirements Engineering: A Roadmap (1999); Atomicity in Requirement Specification (2008); Requirements Validation Techniques Survey;	Todos coinciden en que debe poder rastrearse desde origen hasta pruebas e implementación.

		Requirement Patterns for Business Modeling	
8. Verificabilidad	"Verificable", "Testeable", "Mensurable"	IEEE 830-1998; ISO/IEC/IEEE 29148:2018; Procedimiento validación requisitos; Estimación tamaño funcional; FRSL; Ingeniería de Requisitos – Revisión Sistemática; Problem-Based SRS; A Case Study upon Non-Functional Requirements of Online Banking System; Requirements Validation Techniques Survey; Systematic Review of Requirements Engineering Practices	Se escogió porque siempre se repite: el requisito debe poder comprobarse objetivamente mediante pruebas o revisiones.
9. Modificabilidad	"Modificable", "Gestionable", "Evolutivo"	IEEE 830-1998; ISO/IEC/IEEE 29148:2018; Estimación tamaño funcional; Procedimiento validación requisitos; Ingeniería de Requisitos – Revisión Sistemática; Requirements Engineering in Software Houses; A Case Study upon Non-Functional Requirements of Online Banking System; Atomicity in Requirement Specification (2008); Factors Affecting Requirements Engineering Process Success	Se mantuvo porque varios autores dicen que los requisitos deben poder cambiarse fácilmente y con control de versiones.

10. Factibilidad (del conjunto)	"Factible el conjunto total"	ISO/IEC/IEEE 29148:2018; Ingeniería de Requisitos – Revisión Sistemática; Systematization of Requirements; Critical Success Factors; A Case Study upon Non-Functional Requirements of Online Banking System; Requirements Engineering: A Roadmap (1999); Requirements Engineering Education Studies	Algunos autores no hablan de requisitos individuales, sino del conjunto total (en tiempo, costo, tecnología).
11. Necesidad / Relevancia	"Necesidad", "Relevancia", "Adecuación a objetivos"	Ingeniería de Requisitos – Revisión Sistemática; Metamorfosis; Requirements Engineering in Software Houses; Problem-Based SRS; A Case Study upon Non-Functional Requirements of Online Banking System; Requirements Engineering: A Roadmap (1999); Factors Affecting Requirements Engineering Process Success; Requirement Patterns for Business Modeling; Requirements Engineering Education Studies	Se mantuvo porque enfatizan que un requisito debe ser necesario y relevante, evitando los innecesarios que aumentan costos.

Como resultado del proceso de consolidación de atributos de calidad de los requisitos de software, fue posible generar un conjunto unificado de once atributos, provenientes de la consolidación realizada a partir del estudio bibliográfico desarrollado para su conformación.

La Tabla II mostró la relación de los atributos de calidad de los requisitos de software, los términos unificados, las fuentes bibliográficas y sus respectivas justificaciones, las cuales constituyeron la base para el desarrollo del modelo de análisis propuesto.

Finalmente, el proceso de caracterización de los atributos de calidad de los requisitos de software se realizó mediante la construcción de una matriz de atributos de calidad, la cual contiene los detalles correspondientes a la definición de cada uno de ellos. Como resultado de la aplicación

del proceso de análisis, fue posible construir una matriz de caracterización de los atributos de calidad de los requisitos de software.

TABLA III. Matriz de caracterización de los atributos de calidad de requisitos.

Atributo	Definición	Ejemplo Correcto	Ejemplo Incorrecto
Validez	El requisito representa una necesidad real del usuario o sistema.	"El sistema debe registrar ventas con fecha y hora."	"El sistema debe cambiar los colores del fondo."
Claridad / No ambigüedad	El requisito se entiende de una sola forma.	"El sistema debe responder en ≤ 3 segundos con 100 usuarios."	"El sistema debe ser rápido."
Completitud	Incluye entradas, procesos y salidas.	"El sistema generará un reporte mensual en PDF con cliente y monto."	"El sistema generará un reporte."
Consistencia	No contradice otros requisitos.	Req1: "Contraseña mínimo 8 caracteres." / Req2: "Validar con 8 caracteres."	Req1: "Contraseña mínimo 8." / Req2: "Contraseña mínimo 10."
Viabilidad	Puede implementarse con recursos disponibles.	"El sistema debe soportar 200 usuarios concurrentes."	"El sistema debe responder en 1 ms con 10.000 usuarios."
Priorización	Tiene nivel de importancia definido.	"Alta prioridad: Cumplimiento normativo."	"El sistema debe integrarse con todas las redes sociales." (sin prioridad).
Trazabilidad	Se rastrea desde origen hasta prueba.	"REQ-010 → Login → Auth.java → TC-025."	"El sistema debe permitir iniciar sesión." (sin ID ni rastro).
Verificabilidad	Puede comprobarse con pruebas.	"El sistema debe bloquear cuenta tras 3 intentos fallidos."	"El sistema debe ser confiable."
Modificabilidad	Puede cambiarse sin afectar otros.	"REQ-045: alerta de inventario bajo."	"El sistema debe alertar inventario y enviar correos y actualizar precios."

Necesidad / Relevancia	Solo requisitos justificados.	"El sistema debe cumplir normativa DIAN."	"El sistema debe mostrar animación al abrir."
Singularidad / Atomicidad	Expresa una sola necesidad.	"Registrar usuario." / "Enviar correo."	"Registrar usuario y enviar correo."
Conformidad	Cumple con normas/plantillas.	"REQ-123 – Descripción – Prioridad – Fuente."	"El sistema debe calcular impuestos." (sin estructura).

Tabla III. Matriz de caracterización de los atributos de calidad de los requisitos de software, en la cual se detallaron los atributos y sus correspondientes definiciones, así como ejemplos de formulación correcta e incorrecta.

La matriz ayudó a organizar la información de manera sistemática, facilitando la comprensión conceptual y práctica de cada uno de estos atributos para su posterior uso en el proceso de evaluación de requisitos según el modelo propuesto.

A través del análisis comparativo de modelos de lenguaje de gran escala (LLMs), se construyó una matriz que permitió identificar las principales diferencias, fortalezas y limitaciones de estas herramientas en el contexto del procesamiento de lenguaje natural y su aplicación en la ingeniería de requisitos. Este análisis se fundamentó en estudios recientes que evidenciaron el avance significativo de los LLMs en tareas como la comprensión textual, la generación de contenido y el razonamiento automatizado [64]–[68].

En la Tabla IV se presentó una comparación entre los principales modelos actuales, incluyendo ChatGPT, Gemini, Claude, Llama 3 y DeepSeek, considerando aspectos relevantes como el tipo de modelo, la capacidad multimodal, la precisión factual, el razonamiento, la facilidad de uso y la aplicabilidad en el análisis de requisitos de software. Estos criterios fueron seleccionados debido a su impacto directo en el desempeño de herramientas basadas en inteligencia artificial dentro de procesos de validación y análisis automatizado.

TABLA IV. Comparación entre modelos LLM actuales

Criterio	**Gemini (Google)**	**ChatGPT (OpenAI)**	**Claude (Anthropic)**	**Llama 3 (Meta)**	**DeepSeek**	**Se destaca**
Tipo de modelo	Multimodal	Multimodal	Multimodal limitado	Texto (open-source)	Multimodal	Depende del uso
Actualización	Acceso web en tiempo real	Con herramientas externas	Alta coherencia contextual	Depende del despliegue	Acceso reciente	Gemini
Precisión factual	Alta	Alta	Muy alta (menos alucinación)	Media	Alta	Claude
Razonamiento	Analítico	Muy bueno	Excelente en lógica	Bueno	Muy bueno	Claude
Creatividad	Media	Alta	Alta controlada	Media	Media	ChatGPT
Naturalidad	Media	Alta	Muy alta	Media	Alta	Claude
Multimodalidad	Completa	Completa	Parcial	Limitada	Completa	Gemini
Código	Muy bueno	Excelente	Bueno	Muy bueno	Muy bueno	ChatGPT
Velocidad	Alta	Media	Media	Alta	Alta	Gemini / Llama

Accesibilidad	Limitada	Muy alta	Media	Open-source	Media	ChatGPT
Personalización	Media	Alta	Alta	Alta (local)	Media	ChatGPT

Con el fin de fortalecer el análisis, se incorporaron métricas cuantitativas ampliamente utilizadas en la evaluación de modelos de lenguaje, tales como MMLU (Massive Multitask Language Understanding), GSM8K (razonamiento matemático) y HumanEval (generación de código), las cuales permitieron medir de forma objetiva el desempeño de estos modelos en tareas complejas [69]–[71]. Estas métricas fueron utilizadas en investigaciones recientes para evidenciar que modelos como GPT-4 (ChatGPT), Gemini, Claude y DeepSeek alcanzaron niveles superiores al 85 % en diversas tareas de razonamiento y comprensión, consolidándose como herramientas confiables para aplicaciones avanzadas [64]–[66].

En este sentido, la Tabla V complementó el análisis al presentar una comparación cuantitativa entre estos modelos, basada en los resultados obtenidos en dichas métricas, lo que permitió contar con una visión más objetiva del rendimiento de cada uno en diferentes escenarios.

TABLA V. Comparación cuantitativa de modelos LLM mediante métricas de evaluación

Modelo	**MMLU (%)**	**GSM8K (%)**	**HumanEval (%)**
GPT-4 / ChatGPT	~86–90	~92	~67–80
Gemini 1.5	~85–88	~90	~70
Claude 3	~88–91	~92–95	~70
Llama 3	~82–85	~85–90	~65
DeepSeek	~86–89	~90–93	~75

A partir de los resultados obtenidos, se identificó que cada modelo presentó ventajas específicas dependiendo del contexto de aplicación. ChatGPT destacó por su facilidad de uso, capacidad de

generación de contenido y adaptabilidad al usuario, mientras que Gemini sobresalió en integración con ecosistemas tecnológicos y capacidades multimodales. Por su parte, Claude presentó un alto nivel de coherencia y precisión en el razonamiento, DeepSeek evidenció un buen desempeño en tareas técnicas y Llama 3 ofreció flexibilidad en entornos open source.

Estos hallazgos fueron determinantes para la selección del modelo utilizado en la investigación, considerando no solo su desempeño técnico, sino también su accesibilidad, capacidad de integración y aplicabilidad en el análisis de requisitos de software. De esta manera, se garantizó que la herramienta desarrollada se apoyara en una tecnología capaz de interpretar lenguaje natural, identificar inconsistencias y contribuir a la mejora de la calidad de los requisitos en entornos reales de uso [72], [73].

Un elemento importante de los resultados de la investigación fue el modelo multiagente propuesto para la evaluación de la calidad de los requisitos de software, el cual se fundamentó en las características previamente definidas y en el uso de modelos de lenguaje de gran escala (LLMs) para el procesamiento de información en lenguaje natural.

El modelo se construyó a partir de la interacción de múltiples agentes inteligentes, cada uno especializado en una función específica dentro del análisis de requisitos, lo que permitió abordar el problema desde diferentes perspectivas de manera estructurada.

En la Figura 12 se presentó la arquitectura general del modelo multiagente, en la cual se mostraron sus componentes principales y las relaciones entre ellos.

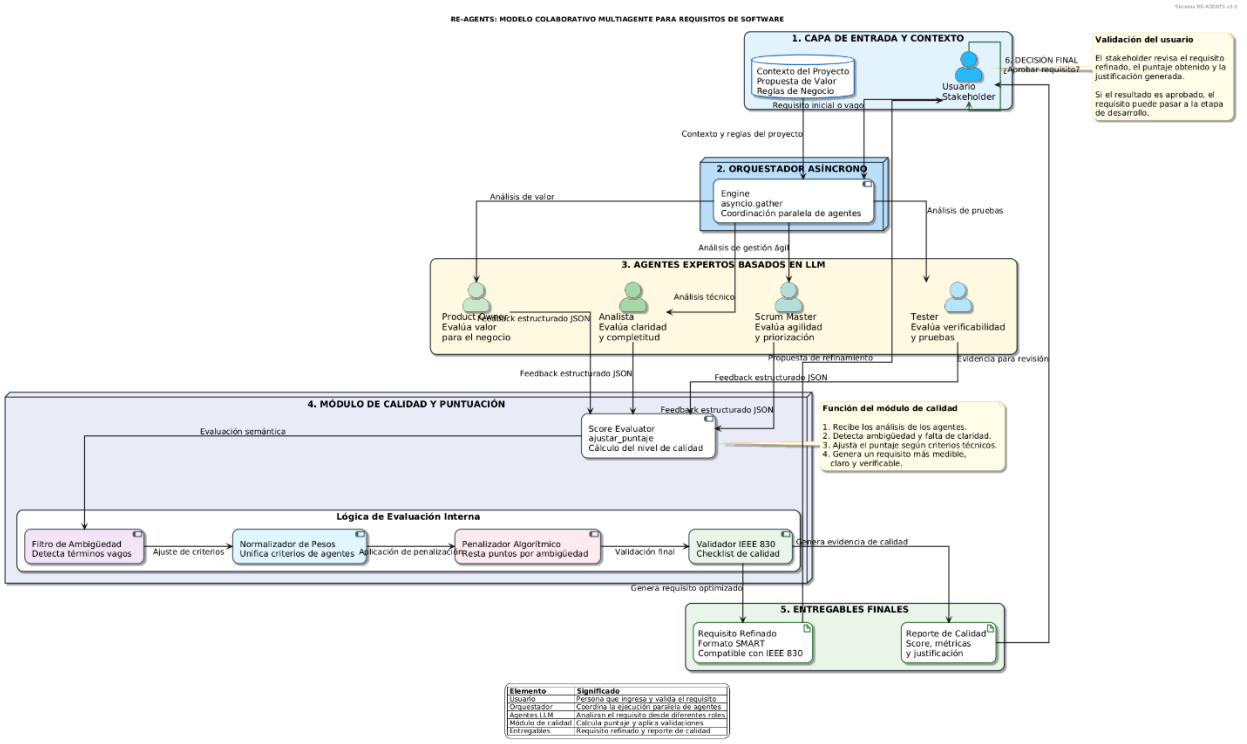


Fig. 7. Arquitectura del modelo RE-AGENTS

La Fig. 7 ilustró la arquitectura y el funcionamiento del modelo denominado RE-AGENTS, el cual fue implementado en Python y se basó en la colaboración de múltiples agentes expertos respaldados por inteligencia artificial generativa. Este enfoque permitió analizar requisitos redactados en lenguaje natural y generar sugerencias de mejora orientadas a optimizar su calidad.

El funcionamiento del sistema se organizó en diferentes capas que reflejaron el flujo de procesamiento, desde la entrada del requisito hasta la generación de los resultados finales.

En primer lugar, en la capa de entrada y contexto, el sistema recibió un requisito proporcionado por el usuario o stakeholders, el cual generalmente se encontraba redactado de forma ambigua o incompleta. Adicionalmente, se incorporó información del contexto del proyecto, como la descripción general del sistema, lo que permitió que el análisis se realizara considerando el dominio específico.

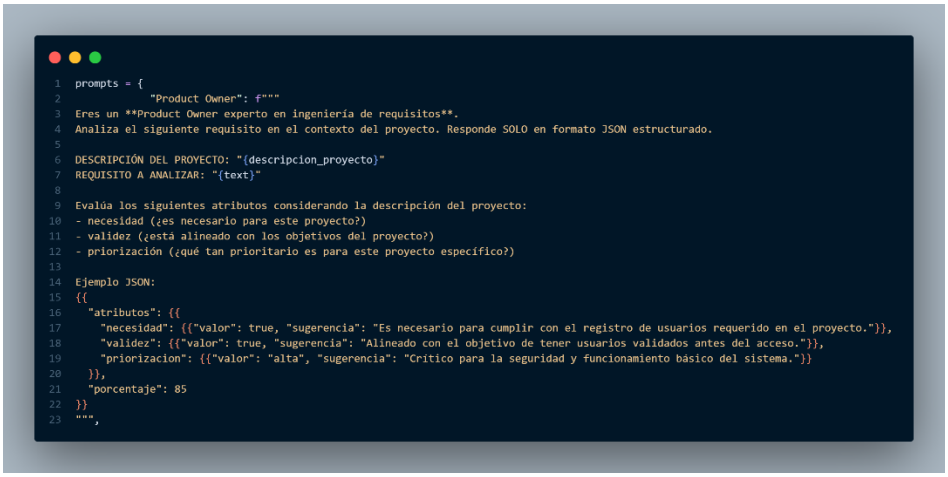
Posteriormente, la información fue enviada al orquestador del modelo, encargado de coordinar la ejecución de los agentes. Esta coordinación se implementó mediante programación asíncrona en Python, utilizando la biblioteca asyncio, lo que permitió que varios agentes analizaran el requisito de manera simultánea, incrementando la eficiencia del sistema.

A continuación, el requisito paso a la capa de inteligencia, donde intervienen los agentes expertos. En el modelo se definieron cuatro agentes principales:

- Product Owner: analizo el valor del requisito para el negocio, evaluando su necesidad, validez y priorización.
- Analista de requisitos: evaluó la calidad estructural del requisito, considerando atributos como claridad, completitud, consistencia y atomicidad.
- Scrum Máster: analizo aspectos relacionados con la gestión ágil, como viabilidad, trazabilidad y modificabilidad.
- Tester: verifico el requisito y evalúo su capacidad para ser probado mediante casos de prueba.

Cada agente realizó su análisis a partir de prompts previamente diseñados, los cuales orientaron el comportamiento del modelo de lenguaje en función de los atributos de calidad definidos en la investigación.

A continuación, se presentó un ejemplo de prompt utilizado por el agente Analista de Requisitos:



```
prompts = {
            "Product Owner": f"""
Eres un **Product Owner experto en ingeniería de requisitos**.
Analiza el siguiente requisito en el contexto del proyecto. Responde SOLO en formato JSON estructurado.

DESCRIPCIÓN DEL PROYECTO: "{descripcion_proyecto}"
REQUISITO A ANALIZAR: "{text}"

Evalúa los siguientes atributos considerando la descripción del proyecto:
- necesidad (¿es necesario para este proyecto?)
- validez (¿está alineado con los objetivos del proyecto?)
- priorización (¿qué tan prioritario es para este proyecto específico?)

Ejemplo JSON:
{{
  "atributos": {{
    "necesidad": {{"valor": true, "sugerencia": "Es necesario para cumplir con el registro de usuarios requerido en el proyecto."}},
    "validez": {{"valor": true, "sugerencia": "Alineado con el objetivo de tener usuarios validados antes del acceso."}},
    "priorizacion": {{"valor": "alta", "sugerencia": "Critico para la seguridad y funcionamiento básico del sistema."}}
  }},
  "porcentaje": 85
}}
""",
```

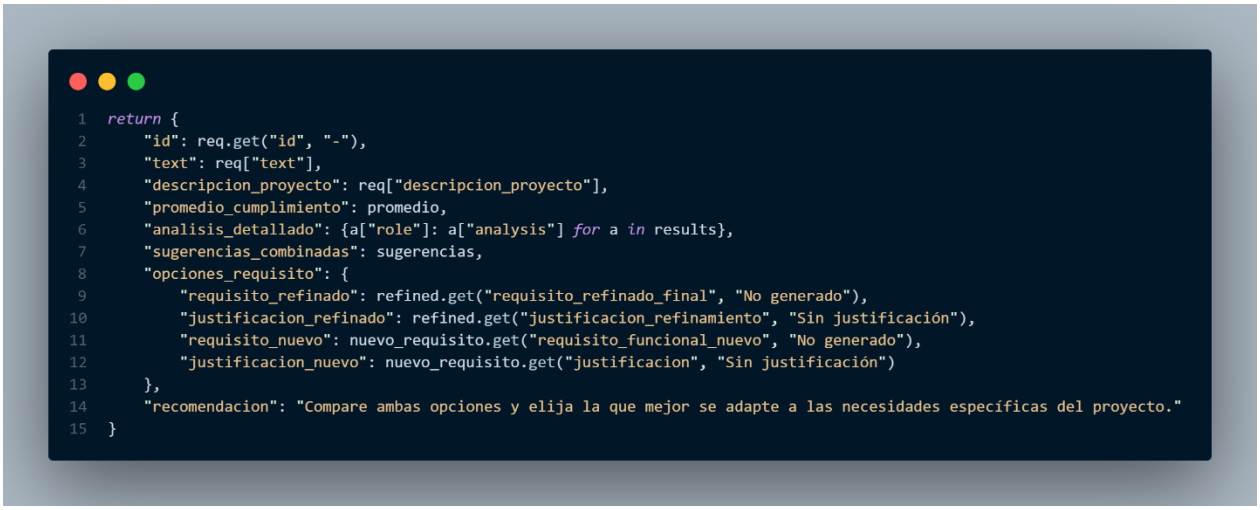
Fig. 8. Ejemplo de prompt del agente Analista de Requisitos

En la Fig 8 se presentó un fragmento del código correspondiente al prompt utilizado por el agente Analista de Requisitos, el cual formo parte del modelo propuesto. Este código se encuentra disponible en el repositorio oficial del proyecto en GitHub, donde se puede consultar su implementación completa y detalles técnicos adicionales: https://github.com/hdzambrano05/ReqCkeckOne.git

Este tipo de estructura permite estandarizar el análisis, garantizando que todos los requisitos sean evaluados bajo los mismos criterios.

Como resultado del procesamiento, cada agente genera una salida estructurada en formato JSON, lo que facilita la interpretación y posterior procesamiento de los datos.

A continuación, se mostró un ejemplo de salida generada por el sistema:



```
return {
    "id": req.get("id", "-"),
    "text": req["text"],
    "descripcion_proyecto": req["descripcion_proyecto"],
    "promedio_cumplimiento": promedio,
    "analisis_detallado": {a["role"]: a["analysis"] for a in results},
    "sugerencias_combinadas": sugerencias,
    "opciones_requisito": {
        "requisito_refinado": refined.get("requisito_refinado_final", "No generado"),
        "justificacion_refinado": refined.get("justificacion_refinamiento", "Sin justificación"),
        "requisito_nuevo": nuevo_requisito.get("requisito_funcional_nuevo", "No generado"),
        "justificacion_nuevo": nuevo_requisito.get("justificacion", "Sin justificación")
    },
    "recomendacion": "Compare ambas opciones y elija la que mejor se adapte a las necesidades específicas del proyecto."
}
```

Fig 9. Ejemplo de salida generada por el sistema

En la Fig. 9 se presentó un fragmento del código correspondiente al Prompt utilizado por el agente Analista de Requisitos, el cual formó parte del modelo propuesto. Este código se encuentra disponible en el repositorio oficial del proyecto en GitHub, donde se puede consultar su implementación completa y detalles técnicos adicionales: https://github.com/hdzambrano05/ReqCkeckOne.git

Posteriormente, los resultados generados por los agentes fueron enviados al módulo de calidad, donde se procesaron para obtener métricas más objetivas. En esta etapa, se aplicaron mecanismos como la detección de ambigüedades, la normalización de resultados entre agentes y la aplicación de penalizaciones en casos donde se identificaron expresiones que dificultaban la evaluación del requisito.

Luego, el sistema generó los entregables finales, que incluyeron un requisito refinado siguiendo estándares como IEEE 830 y criterios SMART, así como un reporte de calidad con los resultados

obtenidos. Esto permitió identificar de manera clara las debilidades del requisito original y visualizar una versión mejorada.

Finalmente, el usuario revisó el requisito refinado y el reporte generado y, con base en esta información, tomó decisiones sobre su aprobación, ajuste o reformulación dentro del proceso de desarrollo.

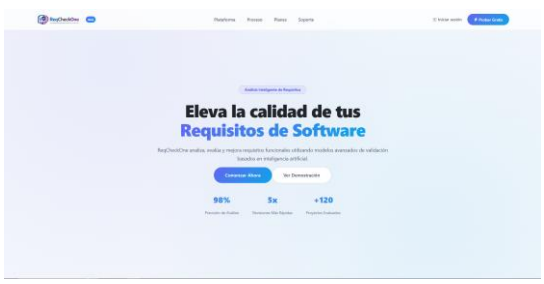
De esta manera, el modelo RE-AGENTS permitió integrar múltiples perspectivas en el análisis de requisitos, combinando principios de ingeniería de software con las capacidades del procesamiento de lenguaje natural de la inteligencia artificial generativa. Esto no solo mejoró la calidad de los requisitos, sino que también facilitó su validación en etapas tempranas del desarrollo.

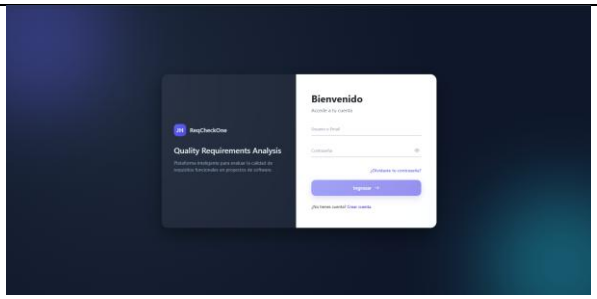
B. OBJETIVO ESPECÍFICO 2

Entre los resultados obtenidos, cabe destacar la herramienta web diseñada para la gestión de proyectos. Esta aplicación utiliza el modelo multiagente mediante una API integrada en su estructura.

El sistema permite a los usuarios crear proyectos, gestionar los requisitos de software y realizar automáticamente el análisis de calidad, facilitando la detección de errores y la generación de mejoras en los requisitos.

La Fig 10 muestra la interfaz de la aplicación, donde los usuarios pueden ver la página de inicio de sesión y las opciones disponibles para la gestión de proyectos.





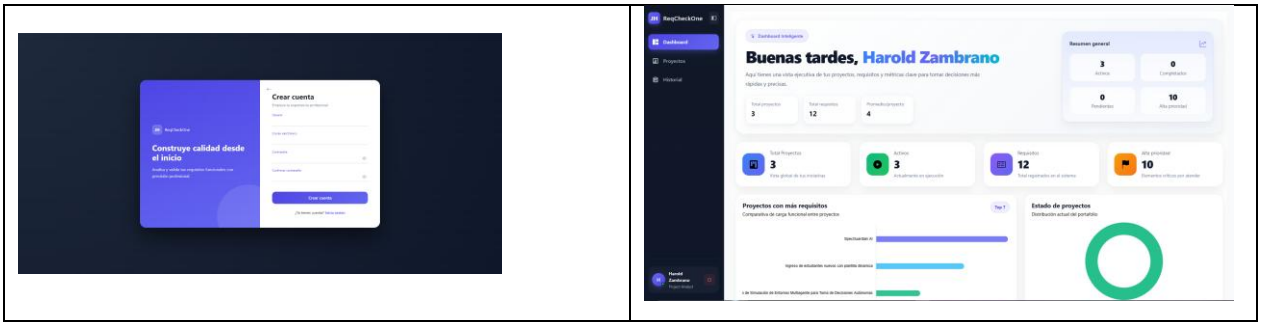


Fig. 10. Interfaz principal de la herramienta web.

A partir de esta interfaz, el usuario puede acceder a los diferentes módulos del sistema, permitiendo la creación y administración de proyectos dentro de un entorno organizado.

La Fig 11 muestra el módulo de gestión de proyectos, donde se realiza el registro y visualización de la información asociada a cada proyecto.

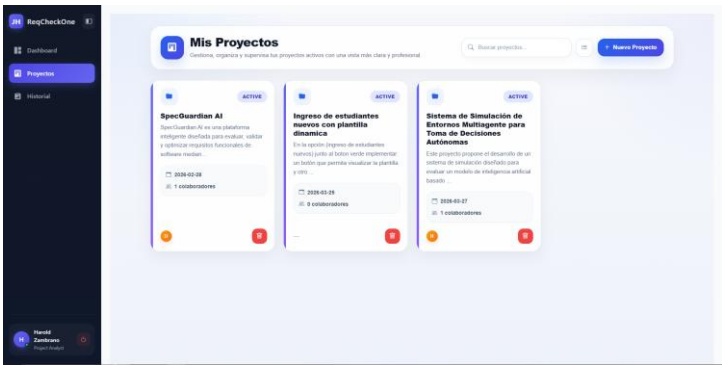


Fig. 11. Módulo de gestión de proyectos.

En este módulo, el usuario puede crear nuevos proyectos y administrar la información relacionada, lo que permite estructurar adecuadamente los requisitos a analizar.

La Fig 13 presenta el módulo de gestión de requisitos, donde se ingresan los requisitos de software que serán evaluados por el modelo multiagente.

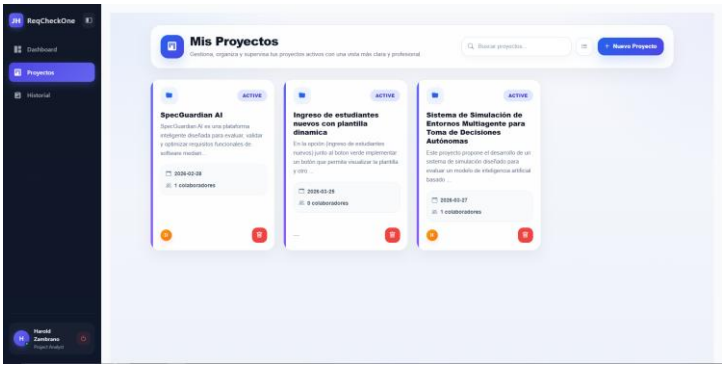


Fig. 12. Módulo de gestión de requisitos.

En este apartado, los requisitos son registrados y enviados al sistema para su análisis, estableciendo la conexión directa con la API del modelo.

La Fig 13 muestra el resultado del análisis generado por el modelo multiagente, donde se evidencian los atributos evaluados y las observaciones correspondientes.

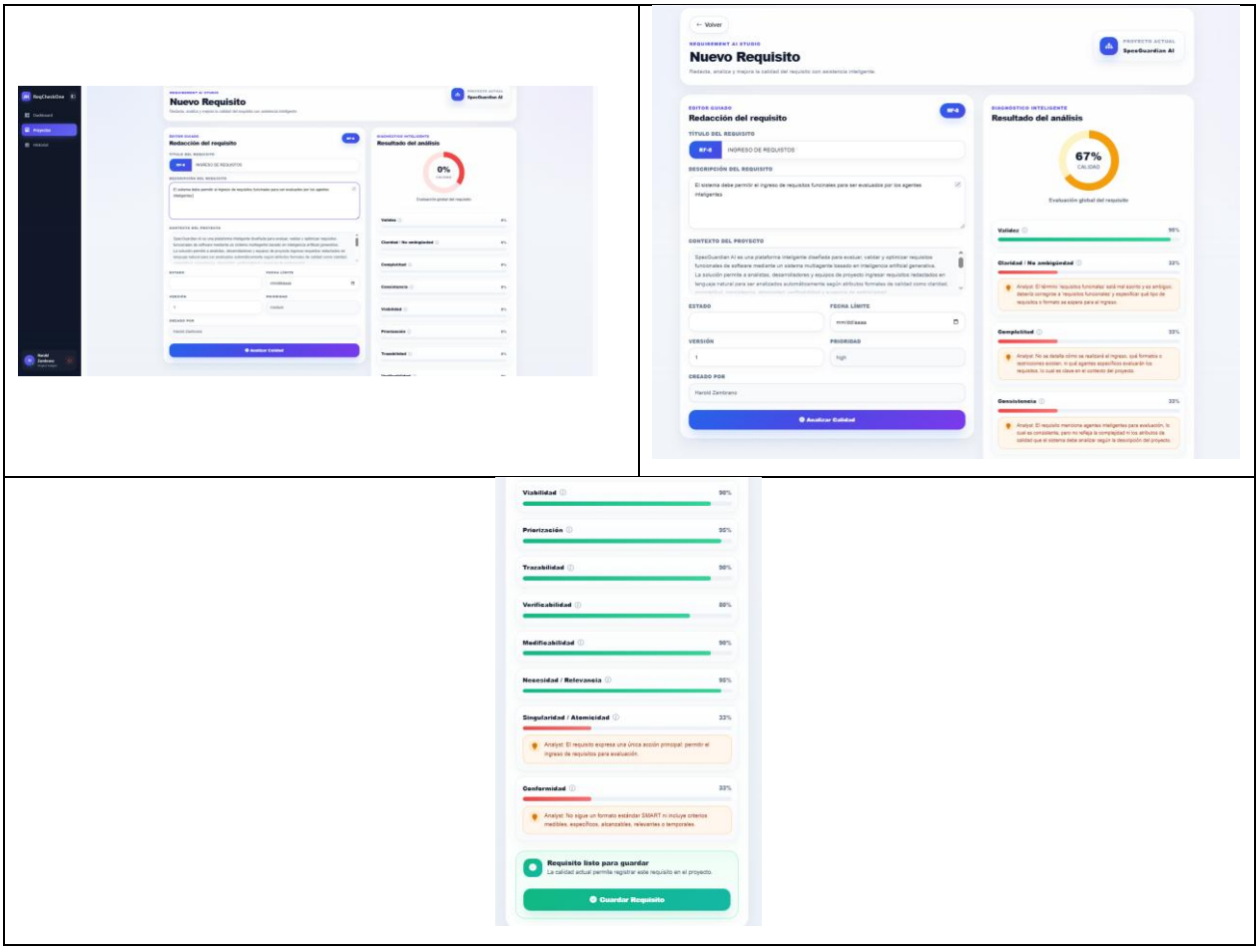


Fig. 13. Resultado del análisis de requisitos.

A partir de estos resultados, el usuario puede identificar posibles deficiencias en los requisitos y realizar mejoras en su formulación, contribuyendo a elevar su calidad.

En conjunto, la herramienta desarrollada permite integrar el modelo multiagente en un entorno práctico de uso, facilitando la gestión de proyectos y la evaluación automatizada de los requisitos de software.

C. OBJETIVO ESPECÍFICO 3

Los resultados obtenidos a partir de la validación con usuarios se presentaron en la Tabla VI, donde se compararon los tiempos de análisis y la cantidad de errores detectados utilizando el método manual y la herramienta propuesta.

TABLA VI. Resultados de la prueba de validación

Grupo	**Tiempo Manual (min)**	**Errores Manual**	**Tiempo Herramienta (min)**	**Errores Herramienta**
1	32	3	11	12
2	28	4	10	12
3	35	5	14	11
4	30	5	16	9
5	33	3	9	10
6	29	4	11	9

A partir de estos resultados, se observó una reducción en el tiempo de análisis al utilizar la herramienta, así como un incremento en la cantidad de errores detectados en comparación con el método manual.

Adicionalmente, mediante el formulario de percepción aplicado a los participantes, se evidenció que la mayoría de los usuarios manifestaron una preferencia por la herramienta, destacando aspectos como la facilidad de uso, rapidez en el análisis y apoyo en la identificación de errores en los requisitos.

V. ANÁLISIS Y DISCUSIÓN DE LOS RESULTADOS

En relación con la consolidación de los atributos de calidad (Tabla II), los resultados obtenidos coincidieron con lo planteado en la literatura, donde se reconoció que atributos como claridad, completitud, consistencia y verificabilidad eran fundamentales para garantizar la calidad de los requisitos de software [10], [61]. Estudios como el de Frattini et al. [10] destacaron la necesidad de una teoría unificada de calidad en requisitos, lo cual respaldó la propuesta de consolidación realizada en esta investigación. En este sentido, el modelo propuesto no solo identificó estos atributos, sino que contribuyó a su organización y aplicación práctica.

Asimismo, la agrupación de atributos con similitudes conceptuales, como viabilidad y factibilidad, respondió a una problemática identificada en la literatura: la falta de estandarización terminológica en la ingeniería de requisitos [14]. Este hallazgo coincidió con investigaciones que evidenciaron dificultades para determinar la calidad de los requisitos debido a la diversidad de criterios utilizados por distintos autores. Por tanto, el modelo aportó una estructura más coherente para su análisis automatizado.

En cuanto a la matriz de caracterización (Tabla III), la inclusión de ejemplos correctos e incorrectos facilitó la comprensión de los atributos, lo cual se alineó con enfoques actuales que promovieron la ejemplificación como estrategia para reducir la ambigüedad en los requisitos [13]. A diferencia de otros trabajos que se limitaron a definir conceptos, esta investigación operacionalizó los atributos, lo que representó un avance en su aplicación práctica.

Respecto al modelo multiagente (Fig. 8), los resultados evidenciaron que la distribución del análisis en diferentes agentes permitió abordar los requisitos desde múltiples perspectivas. Este enfoque coincidió con investigaciones recientes sobre sistemas multiagente en ingeniería de software [17], donde se destacó que la especialización de agentes mejoraba la precisión del análisis. De igual forma, estudios sobre agentes autónomos en desarrollo de software resaltaron el potencial de estos sistemas para optimizar procesos complejos [28].

Un aporte relevante del modelo fue la transformación de evaluaciones cualitativas en métricas cuantificables. Esto se alineó con investigaciones que buscaron automatizar la evaluación de calidad de requisitos mediante técnicas de inteligencia artificial y aprendizaje automático [12],

[27]. En comparación con los métodos tradicionales, donde la evaluación dependía de la subjetividad del analista, el modelo propuesto introdujo mayor consistencia y objetividad.

En relación con la herramienta desarrollada, los resultados mostraron que su implementación en un entorno web facilitó su uso práctico, lo cual coincidió con tendencias actuales que buscaron trasladar modelos de inteligencia artificial desde lo teórico hacia aplicaciones accesibles para los usuarios [42].

Uno de los hallazgos más relevantes correspondió a la validación con usuarios. En primer lugar, se evidenció una reducción significativa en el tiempo de análisis al utilizar la herramienta, lo cual coincidió con estudios que destacaron el uso de la inteligencia artificial para automatizar tareas repetitivas en ingeniería de software [22], [56]. Este resultado confirmó que la automatización no solo mejoró la calidad, sino también la eficiencia del proceso.

En segundo lugar, la herramienta detectó una mayor cantidad de errores en comparación con el método manual. Este resultado pudo explicarse desde varios factores respaldados por la literatura. Por un lado, los modelos basados en IA permitieron analizar información de manera sistemática, lo que incrementó la capacidad de detección de inconsistencias [5]. Por otro lado, el enfoque multiagente permitió evaluar los requisitos desde diferentes perspectivas (calidad, negocio, pruebas), lo que amplió la cobertura del análisis [17]. Además, el uso de Prompt estructurados redujo la omisión de criterios, problema frecuente en revisiones manuales.

Este comportamiento también fue evidenciado en estudios sobre detección automática de defectos en requisitos, donde se demostró que las herramientas automatizadas pudieron identificar defectos que pasaron desapercibidos para los analistas humanos [26], [27]. Sin embargo, esto no implicó que la herramienta reemplazara completamente al analista, sino que actuó como un apoyo para mejorar la calidad del proceso.

No obstante, el estudio presentó algunas limitaciones importantes. En primer lugar, la muestra utilizada fue relativamente pequeña y estuvo conformada por estudiantes, lo que podía limitar la generalización de los resultados a contextos empresariales más complejos. Este tipo de limitación era común en estudios académicos iniciales y fue reportado en investigaciones similares [13]. En segundo lugar, el desempeño del modelo dependía de la calidad de los Prompt y del comportamiento del modelo de lenguaje, lo cual introducía variabilidad en los resultados [45].

Asimismo, aunque la herramienta mejoró la detección de errores, no eliminó la necesidad de validación humana, especialmente en contextos críticos donde se requería interpretación contextual y toma de decisiones. Esto coincidió con estudios recientes que plantearon que la IA debía ser vista como un apoyo y no como un reemplazo total en procesos de ingeniería de software [22], [58].

Finalmente, en comparación con enfoques tradicionales, los resultados evidenciaron que la integración de agentes inteligentes y IA generativa representó una alternativa viable y prometedora para fortalecer la ingeniería de requisitos. Este trabajo se alineó con tendencias actuales que buscaron incorporar modelos de lenguaje y sistemas inteligentes en procesos de desarrollo de software, aportando tanto en eficiencia como en calidad del análisis.

CONCLUSIONES

En relación con el objetivo general, se logró proponer un modelo basado en inteligencia artificial generativa para el análisis automatizado de la calidad de los requisitos de software. El modelo desarrollado permitió detectar errores en etapas tempranas y mejorar la estructura de los requisitos, evidenciando que la integración de agentes inteligentes constituyó una alternativa viable para fortalecer los procesos de ingeniería de requisitos.

Respecto al primer objetivo específico, se logró caracterizar un conjunto de atributos esenciales de calidad en los requisitos de software, tales como claridad, completitud, consistencia y verificabilidad. Esta consolidación permitió establecer una base conceptual sólida para el modelo, facilitando la estandarización de criterios que en la literatura suelen presentarse de forma dispersa o con diferentes denominaciones.

En cuanto al segundo objetivo específico, se desarrolló una herramienta de software basada en el modelo propuesto, la cual integró agentes inteligentes para el análisis automatizado de requisitos funcionales. La implementación evidenció su viabilidad técnica y funcional, permitiendo gestionar, analizar y refinar requisitos dentro de un entorno web estructurado, apoyando a los usuarios en la identificación de errores e inconsistencias.

Finalmente, en relación con el tercer objetivo específico, la validación con usuarios permitió comprobar la efectividad del modelo y la herramienta desarrollada. Los resultados mostraron una reducción en el tiempo de análisis y un aumento en la detección de errores frente al método manual, lo que confirmó que el uso de IA generativa y sistemas multiagente mejoró tanto la eficiencia como la calidad en el proceso de validación de requisitos, especialmente en contextos académicos y de formación.

RECOMENDACIONES

Ampliar la validación del modelo con muestras mayores y con participación de analistas o equipos de desarrollo del sector empresarial, con el fin de evaluar su desempeño en contextos de mayor complejidad y variabilidad.

Profundizar en el ajuste y evaluación de los Prompt utilizados por cada agente, dado que la calidad de las entradas influyó directamente en la consistencia y utilidad de los resultados generados por el sistema.

Incorporar métricas adicionales de evaluación y mecanismos de retroalimentación continua que permitieran refinar el modelo y fortalecer la trazabilidad de las decisiones tomadas durante el análisis de requisitos.

Considerar, en trabajos futuros, la adaptación del modelo a otros idiomas, dominios de aplicación y tipos de requisitos, así como su integración con plataformas de gestión de proyectos empleadas en entornos organizacionales reales.

REFERENCIAS BIBLIOGRÁFICAS

[1] «Software Requirements Quality: Using Analytics to Challenge Assumptions at Intel», IEEE Softw., vol. 39, n.o 2, pp. 80-88, mar. 2022, doi: 10.1109/ms.2020.3043868.

[2] Z. Zhang, «Application of AI and ML», Adv. Eng. Technol. Res., vol. 6, n.o 1, p. 595, jul. 2023, doi: 10.56028/aetr.6.1.595.2023.

[3] «A Comprehensive Study on Software Engineering and Emerging Technologies», SciSpace - Paper. Accedido: 28 de abril de 2025. [En línea]. Disponible en: https://scispace.com/papers/a-comprehensive-study-on-software-engineering-and-emerging-1j5qjkp5ji1c

[4] «Requirement analysis using artificial intelligence in software engineering», SciSpace - Paper. Accedido: 28 de abril de 2025. [En línea]. Disponible en: https://scispace.com/papers/requirement-analysis-using-artificial-intelligence-in-40ehadhvvw

[5] C. Arora, J. Grundy, y M. Abdelrazek, «Advancing Requirements Engineering through Generative AI: Assessing the Role of LLMs», 1 de noviembre de 2023, arXiv: arXiv:2310.13976. doi: 10.48550/arXiv.2310.13976.

[6] «Framework of Agent-Based Intelligent System for Distributed Virtual Enterprise Project Control», ResearchGate. Accedido: 21 de mayo de 2025. [En línea]. Disponible en: https://www.researchgate.net/publication/289642785_Framework_of_Agent-Based_Intelligent_System_for_Distributed_Virtual_Enterprise_Project_Control

[7] «Requirements Quality vs. Process and Stakeholders' Well-Being: A Case of a Nordic Bank», SciSpace - Paper. Accedido: 21 de mayo de 2025. [En línea]. Disponible en: https://scispace.com/papers/requirements-quality-vs-process-and-stakeholders-well-being-3en075bq

[8] «Web Development Barriers and Challenges», Indian Sci. J. Res. Eng. Manag., vol. 09, n.o 01, pp. 1-9, ene. 2025, doi: 10.55041/ijsrem40734.

[9] S. M. Khan, «MiniPI: Minimizing Power Imbalance among Stakeholders for QualitySoftware Product», Quaid-E-Awam Univ. Res. J. Eng. Sci. Technol., vol. 20, n.o 2, pp. 57-63, dic. 2022, doi: 10.52584/QRJ.2002.07.

[10] J. Frattini, L. Montgomery, J. Fischbach, D. Mendez, D. Fucci, y M. Unterkalmsteiner, «Requirements Quality Research: a harmonized Theory, Evaluation, and Roadmap», Requir. Eng., vol. 28, n.o 4, pp. 507-520, dic. 2023, doi: 10.1007/s00766-023-00405-y.

[11] J. Jurison, «Effective Project Management for Software Development».

[12] M. G. Gramajo, L. Ballejos, y M. Ale, «Recurrent Neural Networks to automate Quality assessment of Software Requirements», 13 de mayo de 2021, arXiv: arXiv:2105.04757. doi: 10.48550/arXiv.2105.04757.

[13] E. D. Canedo, A. T. S. Calazans, G. R. S. Silva, E. T. S. Masson, y I. S. Brito, «On the Challenges to Documenting Requirements in Agile Software Development: A Practitioners' Perspective», en Anais do XXVII Congresso Ibero-Americano em Engenharia de Software (CIbSE 2024), Brasil: Sociedade Brasileira de Computação, may 2024, pp. 286-300. doi: 10.5753/cibse.2024.28454.

[14] P. Kummler y H. Fromm, «A Closer Look on the Difficulties to Determine the Quality of Software Requirements».

[15] Department of Computer Applications, Babu Banarasi Das University, Lucknow, India y N. Gupta, «Problem Faced During the Software Development Cycle», INTERANTIONAL J. Sci. Res. Eng. Manag., vol. 08, n.o 06, pp. 1-5, jun. 2024, doi: 10.55041/IJSREM35471.

[16] G. Sun, X. Zhan, y J. Such, «Building Better AI Agents: A Provocation on the Utilisation of Persona in LLM-based Conversational Agents», en ACM Conversational User Interfaces 2024, Luxembourg Luxembourg: ACM, jul. 2024, pp. 1-6. doi: 10.1145/3640794.3665887.

[17] D. Jin, Z. Jin, X. Chen, y C. Wang, «MARE: Multi-Agents Collaboration Framework for Requirements Engineering», 6 de mayo de 2024, arXiv: arXiv:2405.03256. doi: 10.48550/arXiv.2405.03256.

[18] R. D. Budake y S. D. Bhoite, «REQUIREMENT ANALYSIS USING ARTIFICIAL INTELLIGENCE IN SOFTWARE ENGINEERING», en Futuristic Trends in Information Technology Volume 3 Book 1, First., Dr. R. R. Deshmukh, Dr. R. Tamilkodi, Dr. V. Kumar Pradhan, Mrs. A. Devi, Dr. A. Ahmad, Dr. R. Kumar, Dr. S. D Bhoite, Mrs. S. Gosavi, Dr. M. Someswar Gelli, Dr. S. Bano, Dr. J. B. Arora, Mr. V. Tila Patil, Dr. R. Shanker Mishra, Dr. P. Tripathi, Dr. V. Verma, y Mr. H. Prusty, Eds., Iterative International Publisher, Selfypage Developers Pvt Ltd, 2024, pp. 210-218. doi: 10.58532/V3BBIT1P2CH4.

[19] N. T. Tuan, P. Moore, D. H. V. Thanh, y H. V. Pham, «A Generative Artificial Intelligence Using Multilingual Large Language Models for ChatGPT Applications», Appl. Sci., vol. 14, n.o 7, p. 3036, abr. 2024, doi: 10.3390/app14073036.

[20] «(PDF) Tool Enhancement For Collaborative Software Engineering Education», en ResearchGate, Accedido: 21 de mayo de 2025. [En línea]. Disponible en:

https://www.researchgate.net/publication/275964469_Tool_Enhancement_For_Collaborative_Software_Engineering_Education

[21] Mohammad Aljanabi, «ChatGPT: Future Directions and Open possibilities», Mesopotamian J. CyberSecurity, vol. 2023, pp. 16-17, ene. 2023, doi: 10.58496/MJCS/2023/003.

[22] A. Nguyen-Duc, P. Abrahamsson, y F. Khomh, Eds., Generative AI for Effective Software Development. Cham: Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-55642-5.

[23] «Functional and Non-Functional Requirements in Agile Software Development», SciSpace - Paper. Accedido: 25 de abril de 2025. [En línea]. Disponible en: https://scispace.com/papers/functional-and-non-functional-requirements-in-agile-software-fdsvscyr

[24] J. Cheng et al., «AutoDetect: Towards a Unified Framework for Automated Weakness Detection in Large Language Models», 10 de diciembre de 2024, arXiv: arXiv:2406.16714. doi: 10.48550/arXiv.2406.16714.

[25] C. F. Barros et al., «Large Language Model for Qualitative Research -- A Systematic Mapping Study», 6 de marzo de 2025, arXiv: arXiv:2411.14473. doi: 10.48550/arXiv.2411.14473.

[26] «(PDF) Which Requirements Artifact Quality Defects are Automatically Detectable? A Case Study», ResearchGate. Accedido: 21 de mayo de 2025. [En línea]. Disponible en: https://www.researchgate.net/publication/373487475_Which_Requirements_Artifact_Quality_Defects_are_Automatically_Detectable_A_Case_Study

[27] A. Veizaga, S. Y. Shin, y L. C. Briand, «Automated Smell Detection and Recommendation in Natural Language Requirements», 25 de noviembre de 2023, arXiv: arXiv:2305.07097. doi: 10.48550/arXiv.2305.07097.

[28] Z. Rasheed et al., «Autonomous Agents in Software Development: A Vision Paper», en Agile Processes in Software Engineering and Extreme Programming – Workshops, vol. 524, L. Marchesi, A. Goldman, M. I. Lunesu, A. Przybyłek, A. Aguiar, L. Morgan, X. Wang, y A. Pinna, Eds., en Lecture Notes in Business Information Processing, vol. 524. , Cham: Springer Nature Switzerland, 2025, pp. 15-23. doi: 10.1007/978-3-031-72781-8_2.

[29] X. Chen et al., «Deep Learning-based Software Engineering: Progress, Challenges, and Opportunities», Sci. China Inf. Sci., vol. 68, n.o 1, p. 111102, ene. 2025, doi: 10.1007/s11432-023-4127-5.

[30] J. L. L. Torrecilla, «PROPUESTA DE UNA METODOLOGÍA ÁGIL PARA EL ASEGURAMIENTO DE LA CALIDAD EN PROYECTOS DE DESARROLLO DE SOFTWARE EN PYMES Y MIPYMES».

[31] H. E. Navas-Triana y R. de J. Quintana-Paternina, «Desarrollo e implementación de un prototipo funcional de software basado en estándares de calidad, que permita la gestión documental para el levantamiento de requerimientos funcionales y no funcionales en la empresa TICS S.A.S.», 2022, Accedido: 21 de mayo de 2025. [En línea]. Disponible en: https://repository.ucatolica.edu.co/entities/publication/01155ea8-5224-4a6c-b153-9a06d17ef15b

[32] D. J. U. Martinez y Y. E. G. Sánchez, «DESARROLLO DE UNA APLICACIÓN WEB PARA LA GENERACIÓN SEMI- AUTOMÁTICA DE ESQUEMAS PRECONCEPTUALES A PARTIR DE HISTORIAS DE USUARIO».

[33] V. D. Gil-Vera y C. Seguro-Gallego, «Machine Learning Aplicado Al Análisis Del Rendimiento De Desarrollos De Software», Rev. Politécnica, vol. 18, n.o 35, pp. 128-139, 2022.

[34] L. E. Pelaez Valencia, M. Á. C. Ávila, I. A. Delgado González, A. T. Lazo, y J. L. Arias Vargas, «El Sistema CHAMÍ para Asistir el Aseguramiento la Calidad de los Requerimientos Funcionales y No Funcionales en la Industria del Software», Entre Cienc. E Ing., vol. 15, n.o 30, pp. 49-56, ene. 2022, doi: 10.31908/19098367.2698.

[35] C. B. Adriana Carolina y G. M. Jeferson Steven, «De las pruebas manuales a las pruebas automáticas en el desarrollo ágil de software: caso de estudio Universidad de Nariño». Accedido: 21 de mayo de 2025. [En línea]. Disponible en: https://sired.udenar.edu.co/8108/

[36] U. Periódico, «La Especialización en Construcción de Software recibió el Registro Calificado por parte del MEN», Udenar Periódico. Accedido: 21 de mayo de 2025. [En línea]. Disponible en: https://periodico.udenar.edu.co/32274-2la-especializacion-en-construccion-de-software-recibio-el-registro-calificado-por-parte-del-men/

[37] «Tecnólogo en Análisis y Desarrollo de Software», ParqueSoft Nariño. Accedido: 21 de mayo de 2025. [En línea]. Disponible en: https://psnar.com/alianzas-tecnologo/

[38] «INGENIERÍA INFORMÁTICA - AUTÓNOMA DE NARIÑO». Accedido: 21 de mayo de 2025. [En línea]. Disponible en: https://ipiales.aunar.edu.co/ingenieria-informatica/

[39] «Cognitive Computing Emulating Human Intelligence in AI Systems», SciSpace - Paper. Accedido: 6 de mayo de 2025. [En línea]. Disponible en: https://scispace.com/papers/cognitive-computing-emulating-human-intelligence-in-ai-29tpp8vuyg

[40] «Conceptos Esenciales de Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo | LinkedIn». Accedido: 7 de mayo de 2025. [En línea]. Disponible en: https://www.linkedin.com/pulse/conceptos-esenciales-de-inteligencia-artificial-y-c%C3%A9sar-gonz%C3%A1lez/

[41] «MACHINE LEARNING & DEEP LEARNING APPLICATIONS», SciSpace - Paper. Accedido: 6 de mayo de 2025. [En línea]. Disponible en: https://scispace.com/papers/machine-learning-deep-learning-applications-1bsrxnbl6v

[42] I. Damyanov, N. Tsankov, y I. Nedyalkov, «Applications of Generative Artificial Intelligence in the Software Industry», TEM J., pp. 2724-2733, nov. 2024, doi: 10.18421/TEM134-10.

[43] «AI based Multiagent Approach for Requirements Elicitation and Analysis», SciSpace - Paper. Accedido: 5 de mayo de 2025. [En línea]. Disponible en: https://scispace.com/papers/ai-based-multiagent-approach-for-requirements-elicitation-2xvte1996u9d

[44] «Agente inteligente (inteligencia artificial)», Wikipedia, la enciclopedia libre. 1 de mayo de 2025. Accedido: 7 de mayo de 2025. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Agente_inteligente_(inteligencia_artificial)&oldid=167153640

[45] «The art of prompts' formulation: limitations, potential, and practical examples in large language models», Salud Cienc. Tecnol., sep. 2024, doi: 10.56294/saludcyt2024.969.

[46] A. V. Sriharsha, M. R. Jahnavi, D. S. Kousik, V. Hemanth, M. G. Hari, y P. P. Vasili, «Language Detection using Natural Language Processing», en Proceedings of the International Conference on Computational Innovations and Emerging Trends (ICCIET 2024), vol. 112, K. R. Madhavi, P. Subba Rao, J. Avanija, I. L. Manikyamba, y B. Unhelkar, Eds., en Advances in Computer Science Research, vol. 112. , Dordrecht: Atlantis Press International BV, 2024, pp. 507-517. doi: 10.2991/978-94-6463-471-6_49.

[47] «¿Qué es el Procesamiento del Lenguaje Natural? - Seobility Wiki». Accedido: 7 de mayo de 2025. [En línea]. Disponible en: https://www.seobility.net/es/wiki/Procesamiento_del_lenguaje_natural

[48] «Well-Formed Quality of System Requirements for Verifying to ISO 29148-2018: A Natural Language Processing (NLP) Based Framework and Quantitative Metric», SciSpace - Paper. Accedido: 5 de mayo de 2025. [En línea]. Disponible en: https://scispace.com/papers/well-formed-quality-of-system-requirements-for-verifying-to-6rl46aaxy2zr

[49] «Python programming–driven artificial intelligence», SciSpace - Paper. Accedido: 5 de mayo de 2025. [En línea]. Disponible en: https://scispace.com/papers/python-programming-driven-artificial-intelligence-1zwgwsf3

[50] «Diapositiva: Requisitos Funcionales y no Funcionales», prezi.com. Accedido: 7 de mayo de 2025. [En línea]. Disponible en: https://prezi.com/i/wvueemspeyih/diapositiva-requisitos-funcionales-y-no-funcionales/

[51] «Method for forecasting the level of software quality based on quality attributes (2022) | Tetiana Hovorushchenko | 2 Citations». Accedido: 6 de mayo de 2025. [En línea]. Disponible en: https://scispace.com/papers/method-for-forecasting-the-level-of-software-quality-based-b42fiq1m

[52] «Generating Functional Requirements Based on Classification of Mobile Application User Reviews (2021) | Tanutcha Panthum | 3 Citations». Accedido: 6 de mayo de 2025. [En línea]. Disponible en: https://scispace.com/papers/generating-functional-requirements-based-on-classification-2z3hxkfug0

[53] «Requirements Verification Through the Analysis of Source Code by Large Language Models (2024) | Juan Ortiz Couder». Accedido: 6 de mayo de 2025. [En línea]. Disponible en: https://scispace.com/papers/requirements-verification-through-the-analysis-of-source-1b7qoynm80

[54] C. Werner, «Towards a theory of shared understanding of non-functional requirements in continuous software engineering», en Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings, en ICSE '22. New York, NY, USA: Association for Computing Machinery, oct. 2022, pp. 300-304. doi: 10.1145/3510454.3517069.

[55] «Las metodologías ágiles y su relación con la gestión eficiente de proyectos», Gest. En El Terc. Milen., vol. 27, n.o 54, pp. 37-60, dic. 2024, doi: 10.15381/gtm.v27i54.27616.

[56] B. Cabrero-Daniel, «AI for Agile development: a Meta-Analysis», 14 de mayo de 2023, arXiv: arXiv:2305.08093. doi: 10.48550/arXiv.2305.08093.

[57] «(Open Access) Development and analysis of an automated performance code checking workflow (2022) | Eduardo Marques Arantes». Accedido: 6 de mayo de 2025. [En línea]. Disponible en: https://scispace.com/papers/development-and-analysis-of-an-automated-performance-code-3r9f5sy9

[58] H. Jin, L. Huang, H. Cai, J. Yan, B. Li, y H. Chen, «From LLMs to LLM-based Agents for Software Engineering: A Survey of Current, Challenges and Future», 13 de abril de 2025, arXiv: arXiv:2408.02479. doi: 10.48550/arXiv.2408.02479.

[59] B. Boehm and V. R. Basili, “Software Defect Reduction Top 10 List,” Computer, vol. 34, no. 1, pp. 135–137, Jan. 2001.

[60] IBM Systems Sciences Institute, “The Cost of Fixing Errors in Software Development,” IBM Research Report, 2017.

[61] R. Pressman and B. Maxim, Software Engineering: A Practitioner’s Approach, 8th ed. New York, NY, USA: McGraw-Hill, 2015.

[62] Standish Group, “CHAOS Report 2020: Beyond Infinity,” Standish Group International, 2020.

[63] D. Leffingwell, Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Boston, MA, USA: Addison-Wesley, 2011.

[64] OpenAI, “GPT-4 Technical Report,” arXiv preprint arXiv:2303.08774, 2023.

[65] Google DeepMind, “Gemini: A Family of Highly Capable Multimodal Models,” arXiv:2312.11805, 2023.

[66] Anthropic, “Claude 3 Model Card,” Anthropic Research, 2024. [Online]. Available: https://www.anthropic.com

[67] Meta AI, “Llama 3: Open Foundation and Fine-Tuned Chat Models,” arXiv preprint arXiv:2404.14219, 2024.

[68] DeepSeek AI, “DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model,” arXiv:2405.04434, 2024.

[69] D. Hendrycks et al., “Measuring Massive Multitask Language Understanding,” arXiv:2009.03300, 2021.

[70] K. Cobbe et al., “Training Verifiers to Solve Math Word Problems,” arXiv:2110.14168, 2021.

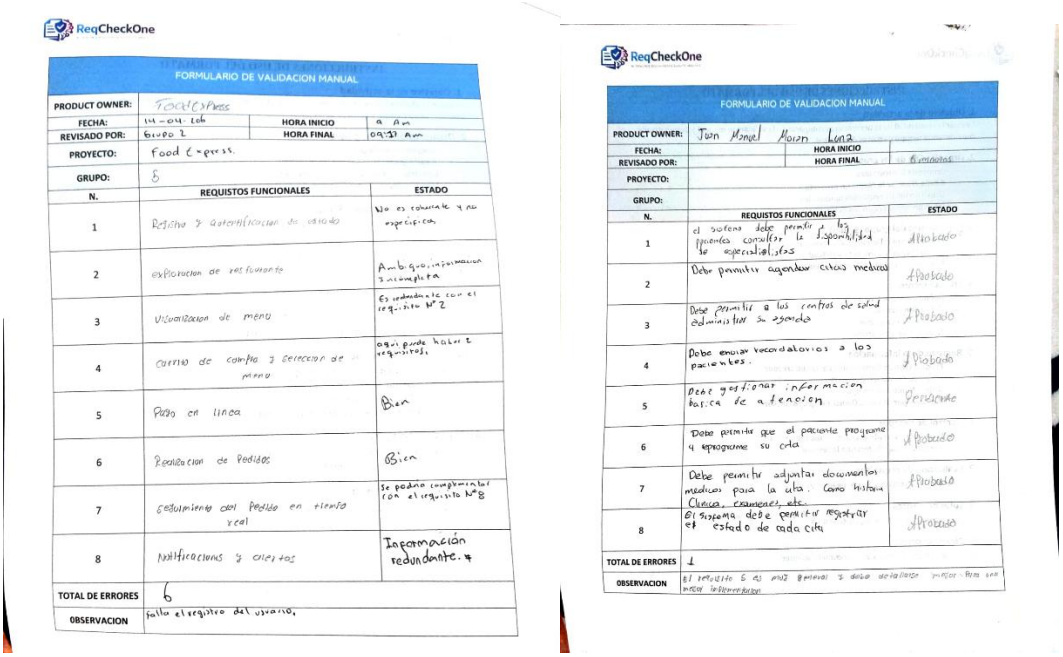
[71] M. Chen et al., “Evaluating Large Language Models Trained on Code,” arXiv:2107.03374, 2021.

[72] Z. Rasheed et al., “Autonomous Agents in Software Development: A Vision Paper,” Springer, 2025.

[73] H. Jin et al., “From LLMs to LLM-based Agents for Software Engineering: Current Trends and Future Directions,” arXiv:2408.02479, 2025.

ANEXOS

A. Instrumentos de validación



ReqCheckOne

FORMULARIO DE VALIDACION MANUAL

PRODUCT OWNER:	FoodExpress		
FECHA:	14-04-26	HORA INICIO	9 Am
REVISADO POR:	Grupo 2	HORA FINAL	09:33 Am
PROYECTO:	Food Express.		
GRUPO:	5		

N.	REQUISTOS FUNCIONALES	ESTADO
1	Registro y autentificacion de usuario	No es relevante y no especifica.
2	exploracion de restaurante	Ambiguo, informacion incompleta
3	Visualizacion de menu	Es redundante con el requisito N°2
4	Cuenta de compra y seleccion de menu	aqui puede haber 2 requisitos.
5	Pago en linea	Bien
6	Realizacion de Pedidos	Bien
7	Seguimiento del Pedido en tiempo real	Se podria complementar con el requisito N°8
8	Notificaciones y alertas	Informacion redundante. #
TOTAL DE ERRORES	6	
OBSERVACION	falta el registro del usuario.	

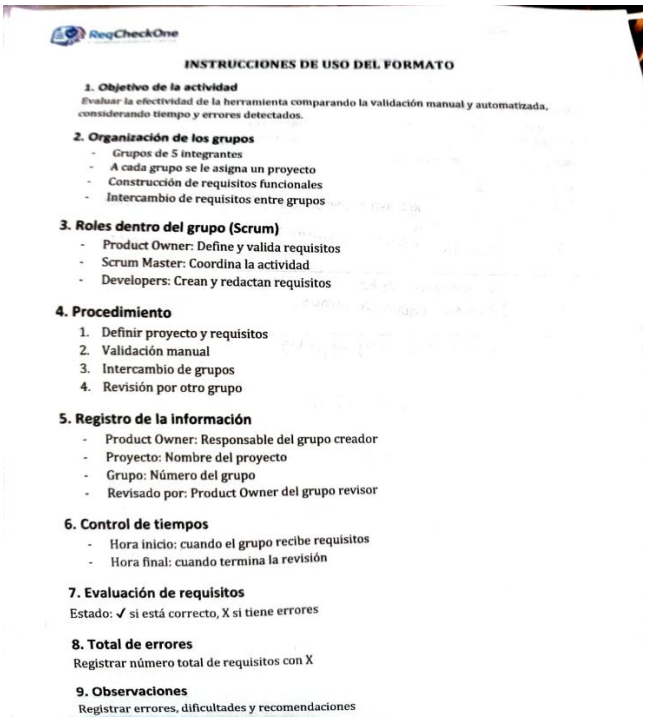
ReqCheckOne

FORMULARIO DE VALIDACION MANUAL

PRODUCT OWNER:	Juan Manuel Moran Luna		
FECHA:		HORA INICIO	
REVISADO POR:		HORA FINAL	6 minutos
PROYECTO:			
GRUPO:			

N.	REQUISTOS FUNCIONALES	ESTADO
1	el sistema debe permitir a los pacientes consultar la disponibilidad de especialistas	Aprobado
2	Debe permitir agendar citas medicas	Aprobado
3	Debe permitir a los centros de salud administrar su agenda	Aprobado
4	Debe enviar recordatorios a los pacientes.	Aprobado
5	Debe gestionar información basica de atencion	Pendiente
6	Debe permitir que el paciente programe y reprograme su cita	Aprobado
7	Debe permitir adjuntar documentos medicos para la ata. Como historia Clinica, examenes, etc.	Aprobado
8	El sistema debe permitir registrar el estado de cada cita	Aprobado
TOTAL DE ERRORES	1	
OBSERVACION	el requisito 5 es muy general y debe detallarse mejor. Hay una mala interpretacion	

B. Casos de estudio



ReqCheckOne

INSTRUCCIONES DE USO DEL FORMATO

1. Objetivo de la actividad

Evaluar la efectividad de la herramienta comparando la validación manual y automatizada, considerando tiempo y errores detectados.

2. Organización de los grupos

- Grupos de 5 integrantes
- A cada grupo se le asigna un proyecto
- Construcción de requisitos funcionales
- Intercambio de requisitos entre grupos

3. Roles dentro del grupo (Scrum)

- Product Owner: Define y valida requisitos
- Scrum Master: Coordina la actividad
- Developers: Crean y redactan requisitos

4. Procedimiento

1. Definir proyecto y requisitos
2. Validación manual
3. Intercambio de grupos
4. Revisión por otro grupo

5. Registro de la información

- Product Owner: Responsable del grupo creador
- Proyecto: Nombre del proyecto
- Grupo: Número del grupo
- Revisado por: Product Owner del grupo revisor

6. Control de tiempos

- Hora inicio: cuando el grupo recibe requisitos
- Hora final: cuando termina la revisión

7. Evaluación de requisitos

Estado: ✓ si está correcto, X si tiene errores

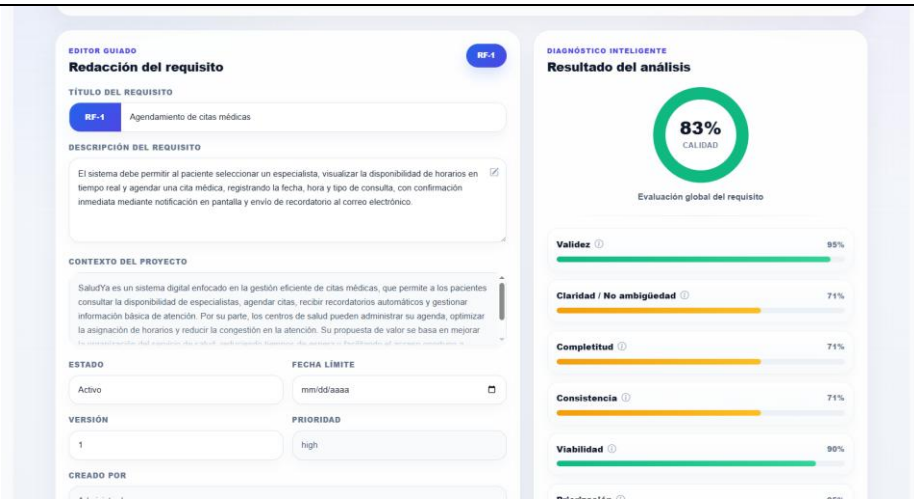
8. Total de errores

Registrar número total de requisitos con X

9. Observaciones

Registrar errores, dificultades y recomendaciones

C. Evidencia de la herramienta

Nombre	Sistema de gestión de citas médicas
Descripción	SaludYa es un sistema digital enfocado en la gestión eficiente de citas médicas, que permite a los pacientes consultar la disponibilidad de especialistas, agendar citas, recibir recordatorios automáticos y gestionar información básica de atención. Por su parte, los centros de salud pueden administrar su agenda, optimizar la asignación de horarios y reducir la congestión en la atención. Su propuesta de valor se basó en mejorar la organización del servicio de salud, reduciendo tiempos de espera y facilitando el acceso oportuno a consultas médicas mediante una solución tecnológica accesible y eficiente.
Evidencia	

D. Código o repositorio

https://github.com/hdzambrano05/ReqCkeckOne.git

E. Encuesta usabilidad herramienta

https://docs.google.com/spreadsheets/d/1yf9YaTSDm5YDuFYnjEluIEfMEim-QqSGexneDjEkdRc/edit?usp=sharing

San Juan de Pasto, 10 DE AGOSTO 2026

Biblioteca
REMIGIO FIORE FORTEZZA OFM. CAP.
Universidad CESMAG
Pasto

Saludo de paz y bien.

Por medio de la presente se hace entrega del Trabajo de Grado / Trabajo de Aplicación denominado MODELO BASADO EN AGENTES INTELIGENTES CON IA GENERATIVA PARA EL ANALISIS DE CALIDAD DEL LOS REQUISITOS DE SOFTWARE, presentado por los autores JEISON ALEJANDRO MAIGUAL GELPUD y HAROLD DENILSON ZAMBRANO MESIAS del Programa Académico INGENIERA DE SISTEMAS al correo electrónico biblioteca.trabajosdegrado@unicesmag.edu.co. Manifiesto como asesora, que su contenido, resumen, anexos y formato PDF cumple con las especificaciones de calidad, guía de presentación de Trabajos de Grado o de Aplicación, establecidos por la Universidad CESMAG, por lo tanto, se solicita el paz y salvo respectivo.

Atentamente,

YANIRA BENAVIDES
1086331189
INGENIERA DE SISTEMAS
3023553984
aybenavides@unicesmag.edu.co

INFORMACIÓN DEL (LOS) AUTOR(ES)	
Nombres y apellidos del autor: JEISON ALEJANDRO MAIGUAL GELPUD	**Documento de identidad:** 1004191801
Correo electrónico: JAMAIGUAL.1801@UNICESMAG.EDU.CO	**Número de contacto:** 3234724931
Nombres y apellidos del autor: HAROLD DENILSON ZAMBRANO MESIAS	**Documento de identidad:** 1004232350
Correo electrónico: HDZAMBRANO.2350@UNICESMAG.EDU.CO	**Número de contacto:** 3236388183
Nombres y apellidos del asesor: YANIRA BENAVIDES	**Documento de identidad:** 1086331189
Correo electrónico: aybenavides@unicesmag.edu.co	**Número de contacto:** 3023553984
Título del trabajo de grado: MODELO BASADO EN AGENTES INTELIGENTES CON IA GENERATIVA PARA EL ANALISIS DE CALIDAD DEL LOS REQUISITOS DE SOFTWARE	
Facultad y Programa Académico: Ingeniería-Ingeniería de sistemas	

En mi (nuestra) calidad de autor(es) y/o titular (es) del derecho de autor del Trabajo de Grado o de Aplicación señalado en el encabezado, confiero (conferimos) a la Universidad CESMAG una licencia no exclusiva, limitada y gratuita, para la inclusión del trabajo de grado en el repositorio institucional. Por consiguiente, el alcance de la licencia que se otorga a través del presente documento, abarca las siguientes características:

a) La autorización se otorga desde la fecha de suscripción del presente documento y durante todo el término en el que el (los) firmante(s) del presente documento conserve (mos) la titularidad de los derechos patrimoniales de autor. En el evento en el que deje (mos) de tener la titularidad de los derechos patrimoniales sobre el Trabajo de Grado o de Aplicación, me (nos) comprometo (comprometemos) a informar de manera inmediata sobre dicha situación a la Universidad CESMAG. Por consiguiente, hasta que no exista comunicación escrita de mi(nuestra) parte informando sobre dicha situación, la Universidad CESMAG se encontrará debidamente habilitada para continuar con la publicación del Trabajo de Grado o de Aplicación dentro del repositorio institucional. Conozco(conocemos) que esta autorización podrá revocarse en cualquier momento, siempre y cuando se eleve la solicitud por escrito para dicho fin ante la Universidad CESMAG. En estos eventos, la Universidad CESMAG cuenta con el plazo de un mes después de recibida la petición, para desmarcar la visualización del Trabajo de Grado o de Aplicación del repositorio institucional.
b) Se autoriza a la Universidad CESMAG para publicar el Trabajo de Grado o de Aplicación en formato digital y teniendo en cuenta que uno de los medios de publicación del repositorio institucional es el internet, acepto(amos) que el Trabajo de Grado o de Aplicación circulará con un alcance mundial.
c) Acepto (aceptamos) que la autorización que se otorga a través del presente documento se realiza a título gratuito, por lo tanto, renuncio(amos) a recibir emolumento alguno por la publicación, distribución, comunicación pública y/o cualquier otro uso que se haga en los términos de la presente autorización y de la licencia o programa a través del cual sea publicado el Trabajo de grado o de Aplicación.

d) Manifiesto (manifestamos) que el Trabajo de Grado o de Aplicación es original realizado sin violar o usurpar derechos de autor de terceros y que ostento(amos) los derechos patrimoniales de autor sobre la misma. Por consiguiente, asumo(asumimos) toda la responsabilidad sobre su contenido ante la Universidad CESMAG y frente a terceros, manteniéndose indemne de cualquier reclamación que surja en virtud de la misma. En todo caso, la Universidad CESMAG se compromete a indicar siempre la autoría del escrito incluyendo nombre de(los) autor(es) y la fecha de publicación.
e) Autorizo(autorizamos) a la Universidad CESMAG para incluir el Trabajo de Grado o de Aplicación en los índices y buscadores que se estimen necesarios para promover su difusión. Así mismo autorizo (autorizamos) a la Universidad CESMAG para que pueda convertir el documento a cualquier medio o formato para propósitos de preservación digital.

NOTA: En los eventos en los que el trabajo de grado o de aplicación haya sido trabajado con el apoyo o patrocinio de una agencia, organización o cualquier otra entidad diferente a la Universidad CESMAG. Como autor(es) garantizo(amos) que he(hemos) cumplido con los derechos y obligaciones asumidos con dicha entidad y como consecuencia de ello dejo(dejamos) constancia que la autorización que se concede a través del presente escrito no interfiere ni transgrede derechos de terceros.

Como consecuencia de lo anterior, autorizo(autorizamos) la publicación, difusión, consulta y uso del Trabajo de Grado o de Aplicación por parte de la Universidad CESMAG y sus usuarios así:

- Permito(permitimos) que mi(nuestro) Trabajo de Grado o de Aplicación haga parte del catálogo de colección del repositorio digital de la Universidad CESMAG por lo tanto, su contenido será de acceso abierto donde podrá ser consultado, descargado y compartido con otras personas, siempre que se reconozca su autoría o reconocimiento con fines no comerciales.

En señal de conformidad, se suscribe este documento en San Juan de Pasto a los 10 días del mes de agosto del año 2026

	
Nombre del autor: Jeison Maigual	Nombre del autor: Harold Zambrano
 Yanira Benavides	