Herramienta para la recuperación de datos de puntos de agua en Etiopía usando LLM y arquitectura RAG

Carlos Mario Nasayó Valverde

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

Dirigida por
(MSc) Luisa Fernanda Rincón Pérez

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Maestría en Ingeniería de Software
Santiago de Cali
18 de febrero de 2026

Santiago de Cali, 26 de enero de 2026.

Señores
Pontificia Universidad Javeriana Cali.
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software.
Cali.

Cordial Saludo.

Por medio de la presente hago constar que en mi calidad de director de trabajo de grado he revisado el proyecto titulado "Herramienta para la recuperación de datos de puntos de agua en Etiopía usando LLM y arquitectura RAG" realizado por el estudiante de Magister en Ingeniería de Software Carlos Mario Nasayó Valverde (cod: codigo), el cual se encuentra terminado y considero que cumple con los requisitos para ser sustentado.
Atentamente,

Digitally signed by ef8c01f0-cbc4-401d-b3c8-5a4e76d3f258
DN: CN=ef8c01f0-cbc4-401d-b3c8-5a4e76d3f258
Reason: I am the author of this document
Location:
Date: 2026.01.28 11:48:59-05'00'
Foxit PDF Reader Version: 2024.4.0

(MSc) Luisa Fernanda Rincón Pérez

Santiago de Cali, 18 de febrero de 2026.

Señores
Pontificia Universidad Javeriana Cali
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software
Cali.

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado "Herramienta para la recuperación de datos de puntos de agua en Etiopía usando LLM y arquitectura RAG" con el fin de cumplir con los requisitos exigidos por la Universidad y para que sea sometido a revisión del jurado y cumpla su aprobación, para conseguir posteriormente el título de Magister en Ingeniería de Software.

Atentamente,

Carlos Nasayo

Carlos Mario Nasayó Valverde
Código: 9017322

Ficha Resumen
Trabajo de Grado Maestría en Ingeniería de Software

TíTULO: Herramienta para la recuperación de datos de puntos de agua en Etiopía usando LLM y arquitectura RAG

1. Énfasis: Ingeniería de Software
2. Área de trabajo: Ingenieria de software
3. Tipo de proyecto: Aplicado
4. Estudiante: Carlos Mario Nasayó Valverde
5. Correo electrónico: cnasayo@javerianacali.edu.co
6. Dirección y teléfono: carrera 20 # 55N-310 - 3106070677
7. Director: Luisa Fernanda Rincon Perez
8. Vinculación del director: Vinculación del director: Planta
9. Correo electrónico del director: lfrincon@javerianacali.edu.co
10. Palabras clave: Chatbot, RAG, LLM, Arquitectura de software, Puntos de agua, Etiopía.
11. Fecha de inicio: Julio 15 de 2025
12. Resumen: Este proyecto de grado presenta el desarrollo de una herramienta basada en procesamiento de lenguaje natural que permite a agropastores y extensionistas en Etiopía consultar, en el idioma Ingles, información sobre puntos de agua y su información climática. La iniciativa respondió a la limitada disponibilidad de datos sobre recursos hídricos, situación agravada por el cambio climático, que afecta gravemente a las comunidades rurales del país.

    El objetivo fue diseñar una arquitectura que combinara técnicas de procesamiento de lenguaje natural y recuperación de información, la cual permitió el desarrollo de una herramienta capaz de responder preguntas con datos reales y actualizados sobre los puntos de agua en Etiopía. Además, esta arquitectura facilitó su implementación en el una interfaz conversacional, de manera que la herramienta quedó al alcance de los usuarios a través del canal de comunicación Telegram.

Agradecimientos

En primer lugar, agradezco a Dios, por guiarme y acompañarme a lo largo de este proceso. Asimismo, expreso mi más profundo agradecimiento a mis padres, **Mayerly Valverde** y **Yony Nasayo**, ya que sin su apoyo incondicional, sus enseñanzas y los valores que me han inculcado, no estaría hoy culminando este logro académico.

Agradezco de manera especial a mi directora de trabajo de grado, **Luisa Fernanda Rincón**, por su orientación, acompañamiento y disposición a lo largo del desarrollo de este trabajo. Sus observaciones, conocimientos y compromiso fueron fundamentales para fortalecer el enfoque académico y técnico de la investigación. Asimismo, agradezco al **jurado evaluador** por el tiempo dedicado a la revisión de este trabajo y por sus valiosas observaciones y sugerencias.

Agradezco a **Steven Sotelo** por proporcionarme la base de la idea que dio origen al desarrollo de este trabajo, así como por su apoyo y aporte conceptual durante las etapas iniciales del proyecto.

De manera muy especial, agradezco a mi prometida, **Gabriela Bolaños**, por su apoyo incondicional a lo largo de todo este proceso. Su acompañamiento constante, su paciencia y su respaldo en cada uno de los altibajos que enfrenté fueron fundamentales para poder continuar y culminar este trabajo. Su confianza y apoyo en todo momento fueron un pilar esencial durante este camino académico.

Asimismo, agradezco a mi colega y amigo **Olmer Suárez**, quien confió en mí y me brindó un respaldo económico fundamental al inicio de este proceso, permitiéndome dar los primeros pasos para el desarrollo de este trabajo.

Resumen

Este proyecto de grado presenta el desarrollo de una herramienta basada en procesamiento de lenguaje natural que permite a agropastores y extensionistas en Etiopía consultar, en el idioma inglés, información sobre fuentes de agua y pronósticos climáticos. La iniciativa respondio a la limitada disponibilidad de datos sobre recursos hídricos, situación agravada por el cambio climático, que afecta gravemente a las comunidades rurales del país.

El objetivo fue diseñar una arquitectura que combinara técnicas de procesamiento de lenguaje natural y recuperación de información, la cual permitió el desarrollo de una herramienta capaz de responder preguntas con datos reales y actualizados sobre los puntos de agua en Etiopía. Además, esta arquitectura facilitó su implementación en distintos medios tecnológicos, de manera que la herramienta quedó al alcance de los usuarios a través del canal de comunicación Telegram.

Palabras Clave: Chatbot, RAG, LLM, Arquitectura de software, Puntos de agua, Etiopía.

Abstract

This thesis project presents the development of a tool based on natural language processing that enables agropastoralists and extension workers in Ethiopia to consult, in English, information about water sources and climate forecasts. The initiative responded to the limited availability of data on water resources, a situation exacerbated by climate change, which severely affects rural communities in the country.

The objective was to design an architecture that combined natural language processing and information retrieval techniques, which allowed the development of a tool capable of answering questions with real and updated data on water points in Ethiopia. Furthermore, this architecture facilitated its implementation across different technological platforms, making the tool accessible to users through the Telegram communication channel.

Keywords: Natural language, Water monitoring, agro-pastoralists, Decision making, Information retrieval, Software architecture.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Definición del problema

1.1.1. Planteamiento del problema

El acceso limitado a información sobre agua en Etiopía es un problema crítico que afecta a las comunidades que dependen de esto para su sustento, la escasez de lluvia y disminución de fuentes de agua solo en el periodo de 2021 y 2022 provocó la pérdida de más de 6 millones de cabezas de ganado (Dr. Endrias Geta, citado en (Alemayehu et al., 2024a)), lo que no solo causa escasez alimenticia sino que también genera conflictos entre las diferentes comunidades étnicas (Alemayehu, 2022). Además, las fuentes de agua destinadas al uso doméstico están desapareciendo, obligando a los pastores a compartir el agua con el ganado(Alemayehu and Workeneh, 2023), lo que agrava aún más la situación.

Si bien se han desarrollado herramientas para el monitoreo de puntos de agua, su uso sigue siendo limitado debido a factores como la alfabetización digital y la falta de diversidad de canales para consultar esta información más allá de aplicaciones web. Como resultado, muchos agropastores y extensionistas no pueden beneficiarse de estos datos, aun cuando están disponibles. Esto lleva a que continúen prácticas como desplazarse hacia puntos de agua conocidos sin certeza de que en ese momento haya agua disponible, lo que conlleva pérdidas significativas de tiempo y esfuerzo. En la **En la Figura 1.1 se puede observar el árbol del problema.**

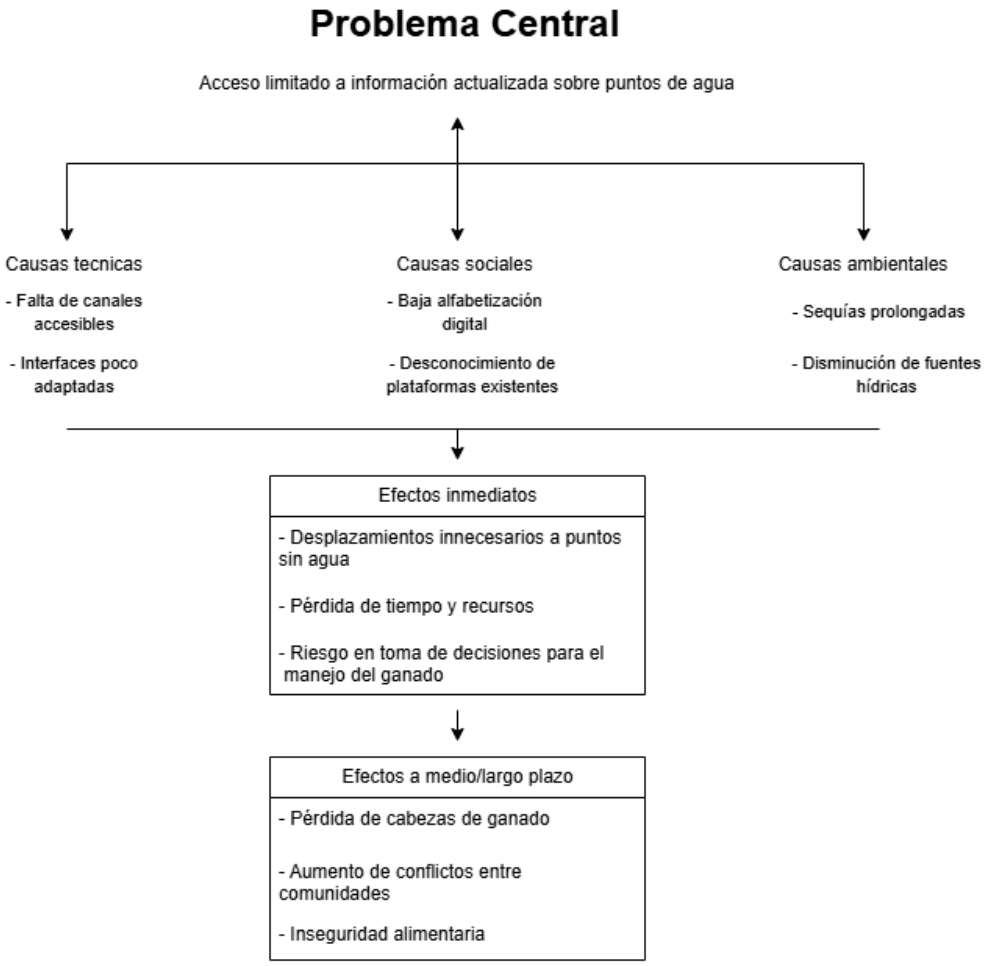


Figura 1.1: Árbol del problema
Nota: Figura de elaboración propia .

Desde la revolución grandes modelos de lenguaje de gran tamaño (LLM por sus siglas en inglés) en 2022, estas tecnologías han surgido como una alternativa para facilitar la interacción entre personas y sistemas de información. Su integración con herramientas conversacionales como Telegram, Facebook, WhatsApp o Messenger ha abierto nuevas posibilidades de acceso a datos a través de lenguaje natural. Sin embargo, este enfoque aún no ha sido explorado en el contexto de la información sobre puntos de agua. Por esto, el presente trabajo busca explorar la combinación de LLMs con técnicas de recuperación de información como una alternativa para mejorar la consulta de datos de puntos de agua en Etiopía. Como canal de implementación se plantea el posible uso de Telegram, que se ha convertido en la aplicación de mensajería más descargada en Etiopía(Bahru, 2022). Esto es debido a que muchos Etíopes perciben que en Telegram se permite compartir archivos de gran tamaño como videos, documentos o audiolibros, incluso cuando la conectividad a internet es limitada(Sarkar, 2024).

Ante esta problemática, se hace necesario el desarrollo de herramientas tecnológicas basadas en interfaces conversacionales que, mediante el uso del lenguaje natural, faciliten el acceso a información actualizada sobre el estado de los puntos de agua y los pronósticos climáticos. Por ello, se propone una solución que permita realizar consultas a través de la plataformas de mensajería, siendo Telegram un caso de uso posible por su amplia adopción en Etiopía.

1.1.2. Formulación del problema

Basado en lo anterior, el problema se delimita de la siguiente manera:

¿Cómo diseñar, desarrollar y evaluar una herramienta tecnológica que reconozca el lenguaje natural de los agropastores y extensionistas, y recupere información en fuentes de datos para brindar respuestas contextualizadas sobre información de puntos de agua y pronósticos climáticos en Etiopía?

Esta pregunta tiene las siguientes preguntas derivadas:

¿Cómo se puede garantizar el acceso, almacenamiento y actualización de los datos históricos de puntos de agua en Etiopía?

¿Qué arquitectura permite combinar el procesamiento de lenguaje natural y la recuperación de datos para generar respuestas sobre la disponibilidad de puntos de agua y los pronósticos climáticos?

¿Cómo puede desarrollarse una herramienta que responda preguntas sobre puntos de agua en Etiopía utilizando una arquitectura que combine modelos de lenguaje y recuperación de información?

¿Cómo se puede desplegar esta herramienta en el canal de comunicación Telegram?

¿Qué métricas pueden utilizarse para evaluar la herramienta?

1.2. Objetivos del proyecto

1.2.1. Objetivo General

Diseñar, desarrollar y evaluar una herramienta tecnológica que reconozca el lenguaje natural utilizado por los agropastores y extensionistas, y recupere información en fuentes de datos para brindar respuestas contextualizadas sobre información de puntos de agua y pronósticos climáticos en Etiopía.

1.2.2. Objetivos específicos

1. Diseñar estrategias para el acceso, almacenamiento, actualización y vectorización de los datos históricos de puntos de agua en Etiopía, que serán utilizados por la herramienta para generar respuestas a partir de consultas en lenguaje natural.

2. Diseñar la arquitectura de la herramienta, donde se evidencie la integración de LLM con técnicas de generación aumentada por recuperación (RAG), especificando sus módulos, componentes y la forma en que interactúan entre sí.

3. Desarrollar la herramienta conforme a la arquitectura propuesta.

4. Desplegar la herramienta como una interfaz conversacional a través de un canal de mensajería.

5. Evaluar el desempeño de la herramienta utilizando la métrica de fidelidad (faithfulness) del marco RAGAS, considerando un umbral mínimo de 0.70 como criterio de validez para las respuestas generadas.

1.3. Delimitaciones y alcances

Este trabajo se enfocó en el desarrollo e implementación de una herramienta que permita la recuperación de información sobre puntos de agua y pronósticos climáticos en Etiopía, utilizando un modelo de inteligencia artificial generativa combinado con técnicas de generación aumentada por recuperación (RAG). A continuación, se delimita el alcance del proyecto con base en los objetivos específicos:

1.3.1. Diseño de estrategias para el acceso, almacenamiento y consulta de datos

- Como fuente de datos principal se utilizó una base de datos en MongoDB, la cual contiene información histórica sobre puntos de agua en Etiopía, incluyendo datos climáticos y ubicación geográfica.
- Se definieron estrategias para la transformación y vectorización de los datos almacenados, con el fin de permitir su indexación y recuperación mediante técnicas de generación aumentada por recuperación (RAG).
- Se definió la estrategia para la actualización periódica de la información desde la fuente de datos original hacia la base de datos vectorizada.
- El procesamiento de datos se limitó a información estructurada o semiestructurada proveniente de la base de datos predefinida, sin contemplar el análisis de fuentes no estructuradas como documentos PDF, imágenes o audio.

1.3.2. Diseño de la arquitectura de la herramienta de recuperación de información.

- Se diseñó la arquitectura de la herramienta, en donde se detallaron claramente todos y cada uno de los módulos, componentes o contenedores que harán parte de la solución.
- Se detallaron todas las fuentes de datos utilizadas en el desarrollo de la herramienta.
- La arquitectura estará diseñada para atender consultas y dar respuestas en tiempo real.

1.3.3. Implementación de modelo de LLM con generación aumentada por recuperación

- Se utilizó un modelo LLM pre entrenado y se ajustó con técnicas de generación aumentada.
- Se utilizó Ollama(llama3.1) como modelo LLM local seleccionado.
- El modelo fue capaz de responder consultas en lenguaje natural sobre la disponibilidad de agua, estados e información descriptiva de puntos de agua en Etiopía.
- La herramienta pudo comprender y generar respuestas únicamente en el idioma inglés.
- La herramienta fue capaz de mantener el contexto conversacional, permitiendo responder de forma coherente a preguntas relacionadas con interacciones previas.
- La funcionalidad principal se expuso como un servicio REST, permitiendo la conexión modular con distintos canales de entrada, mediante webhooks o API Gateway.
- La herramienta solo generó respuestas de texto.
- No se implementaron interfaces gráficas avanzadas (web o móviles).
- No se incluyeron funcionalidades de interacción por voz.

1.3.4. Despliegue en un canal de comunicación

- Se implementó la herramienta en un canal de comunicación, usando la red social Telegram.

1.3.5. Evaluación de desempeño de la herramienta

- La evaluación del sistema se llevo a cabo utilizando el marco RAGAS.
- Esta fase de evaluación se realizó exclusivamente mediante pruebas controladas; no se contempla la participación directa de usuarios finales.

1.4. Justificación del trabajo de grado

En Etiopía son muchos los agropastores que dependen de la agricultura y ganadería para su sostenimiento y el de sus familias, por ende dependen en gran medida del agua, que es un recurso vital para el desempeño de esta laborees, pero debido al cambio climático de los últimos años el acceso a este recurso se ha reducido bastante, causando escasez del mismo y afectando así las diferentes prácticas de los agro pastores. Lo cual lleva a los agro pastores a buscar otras fuentes de este recurso para desempeñar sus labores, entre estas tareas se encuentra visitar los diferentes puntos de agua distribuidos en el país.

Sin embargo, esto es una práctica que realizan a ciegas, puesto que no tienen la certeza de si al punto de agua que están yendo tiene o no agua en ese preciso instante, lo que puede derivar

en situaciones como que una persona recorra una larga distancia buscando un punto de agua que conocía, pero en ese momento este seco, lo que les haría perder una cantidad de tiempo considerable.

Si bien existen ya varias herramientas digitales que monitorean estos pozos de agua y dan un estimado del estado en que se encuentran en los últimos días, estas plataformas están enfocadas en personas con conocimiento técnico y en su mayoría son aplicaciones web, lo que hace que su uso por las comunidades de agropastores o extensionistas sea muy baja, puesto que utilizan otros medios como las redes sociales o mensajes de texto.

Ante esta problemática nace la idea de construir una arquitectura sobre la cual se puedan desarrollar herramientas capaces de reconocer el lenguaje natural utilizado por agropastores y extensionistas, y utilizando las fuentes de datos ya existentes, poder generar respuestas con datos reales y recientes de los diferentes puntos de agua, además de incluir un canal de interacción como Telegram. En este contexto, las herramientas de generación y recuperación juegan un papel importante, ya que pueden combinar el reconocimiento de lenguaje, la búsqueda de la información y la generación de una respuesta contextualizada.

Adicionalmente, desde un punto de vista académico, este trabajo aborda un vacío poco explorado en la literatura: la escasa investigación entre modelos LLM de código abierto y modelos comerciales aplicados a dominios agroclimaticos en países de África. Asimismo, la arquitectura propuesta estará diseñada para que pueda ser re utilizable en otros países del cuerno de África como lo son Somalia, Yibuti y Eritrea, que pueden enfrentar situaciones climáticas y sociales similares, lo que incrementa su pertinencia y potencial de escalabilidad.

1.5. Metodología de la investigación

El desarrollo de la herramienta siguió un enfoque Design Science Research (DSR), una metodología orientada al desarrollo y validación de artefactos tecnológicos. Este enfoqué permitió abordar el problema mediante el diseño, implementación y evaluación del artefacto, que en este caso correspondió a la herramienta desarrollada.

Todo este proceso se llevo a cabo de la mano trabajando con una metodología ágil iterativa inspirada en Scrum, organizada en sprints quincenales, que permitieron estructurar el trabajo de forma progresiva y controlada. Cada sprint se enfocó en tareas específicas asociadas a las etapas del ciclo de vida, permitiendo avanzar paso a paso y asegurar la funcionalidad de cada componente de la herramienta.

Adicionalmente, se siguió un pipeline estructurado para el procesamiento de los datos de los puntos de agua, compuesto por las siguientes etapas:

1. **Recolección de los datos**.
2. **Vectorización de los datos**.
3. **Formulación de consultas en lenguaje natural**.
4. **Recuperación contextual y generación de respuestas**.

5. **Actualización periódica del los datos vectorizados**.

6. **Evaluación con RAGAS**.

1.6. Resultados obtenidos

En el presente trabajo se obtuvo una arquitectura de software orientada al desarrollo de una herramienta de recuperación aumentada con generación (RAG) para contestar preguntas en lenguage natural sobre información asociada a puntos de agua en Etiopía. Dicha arquitectura integra múltiples fuentes de datos, combinando el acceso a información climática almacenada en una base de datos (MongoDB) con mecanismos de recuperación semántica sobre una base de datos vectorizada implementada mediante *pgvector*. Esta arquitectura permitió soportar distintos tipos de consultas, desde la obtención directa de registros estructurados hasta preguntas de carácter descriptivo que requieren contextualización.

Como resultado de esta arquitectura, se desarrolló una herramienta capaz de procesar consultas en lenguaje natural y generar respuestas utilizando modelos de lenguaje tanto locales para la detección de intención y limite de dominio como desplegados en la nube para la generación de repuestas y procesamiento de contexto. La herramienta incorpora un flujo de decisión que determina dinámicamente el tipo de recuperación a ejecutar, ya sea mediante consultas directas a bases de datos estructuradas o mediante búsqueda semántica sobre documentos indexados.

A partir de la evaluación de la herramienta se obtuvieron resultados asociados al desempeño del sistema frente a un conjunto de casos de prueba representativos. Dichos resultados incluyen métricas de fidelidad de la respuesta, relevancia con respecto a la pregunta formulada y similitud con una respuesta de referencia, las cuales fueron calculadas para cada uno de los casos evaluados. El análisis detallado de estas métricas y su comportamiento según el tipo de consulta se presenta con mayor profundidad en las secciones posteriores.

Capítulo 2

Marco de referencia

En este capítulo se presentan las bases teóricas y los trabajos previos que sirven como base para la realización de este proyecto.

2.1. Marco Teórico

El enfoque de este trabajo de grado se encuentra en la integración de herramientas de inteligencia artificial, fuentes de datos de recursos puntos de agua en Etiopía, herramientas de monitoreo ya existentes y técnicas de recuperación de información. En esta sección se presentan todos los conceptos de cada una de las áreas mencionadas anteriormente, esto con el fin de establecer referencias solidas que posteriormente van a sustentar la investigación y su implicación en el contexto de combinar herramientas de reconocimiento de lenguaje natural, generación y recuperación de información. La comprensión de estos conceptos resulta fundamental para sustentar la solución propuesta.

2.1.1. Bases Teóricas

En esta sección se describen los fundamentos teóricos que sustentan el trabajo de investigación.

2.1.1.1. Agropastores

Los agropastores en Etiopía, son comunidades cuya subsistencia depende tanto de la ganadería como de el manejo de cultivos. Son poblaciones que se desplazan constantemente en busca de fuentes de agua y pasto, llevando consigo sus familias y ganado que es su mayor sustento. Para estas comunidades, cuidar de su ganado significa cuidar la vida familiar, ya ofrecen productos como leche que es fuente importante de proteína en tiempos de escasez.

Estas comunidades son consideradas como una de las más desfavorecidas en el cuerno de África, enfrentan múltiples desafíos como la sequía, enfermedades y falta de acceso a servicios veterinarios. Estas comunidades han sido relegadas a zonas áridas y muy poco productivas del país.En este contexto, la movilidad y el conocimiento tradicional son fundamentales para su supervivencia.(Nuñez, 2008)

2.1.1.2. Extensionista

El termino extensionista proviene de a palabra extensión, es una educación no formal que tiene como objetivo difundir información, conocimientos prácticos y asesoría, con el fin de mejorar

habilidades, actitudes y aspiraciones de las comunidades, especialmente en contextos rurales, aunque tiende a asociarse más con la agricultura, su aplicación puede extenderse a otros ámbitos de desarrollo.

Un extensionista también puede considerarse un elemento político y organizativo, para facilitar desarrollo con diferentes enfoques que van desde la transferencia de tecnología por parte de empresas organizadas a entornos agrícolas, hasta enfoques educativos de resolución de problemas. Cuando se implementa de forma eficaz, la extensión, y por ende el extensionista contribuyen significativamente al desarrollo social y económico. (Rivera and Qamar, 2003)

2.1.1.3. Puntos de agua en Etiopía

Cuando se mencionan puntos de agua se refiere a los diferentes pozos o almacenamientos de agua que están ubicados en Etiopía, estos puntos abastecen de agua a gran parte de la población, en Waterpoint Data Exchange (2025) se puede observar que existen mas de 100 puntos de agua funcionales en el país, junto con la información sobre la población al rededor de cada punto que utiliza los recursos, las fuentes de donde proviene el agua y la capacidad de almacenamiento que tiene.

2.1.1.4. Inteligencia artificial generativa

La inteligencia artificial generativa se refiere a la capacidad que tiene los sistemas IA para generar contenidos emulando lo que produciría un humano (Corredera, 2023), esta tecnología permite crear contenidos de diferentes tipos como audios, videos, imágenes y textos personalizados en cuestión de segundos y con una simple instrucción.

Algunos de los modelos de IA generativas mas conocidos son GPT-4, DeepSeek y Claude. GPT-4 es un modelo multimodal desarrollado por OpenAI capaz de recibir entradas de texto, documentos e imágenes y con estas entradas producir salidas de texto, a pesar de ser menos capaz que los humanos en varios escenarios, el modelo muestra rendimiento similar al humano en varias pruebas incluso académicas, es un modelo que se basa en Transformer (Nassiri and Akhloufi, 2023) para predecir el siguiente token de un documento (Achiam et al., 2023). Claude es otro modelo de IA desarrollado por Anthropic, tiene un enfoque para ser un asistente conversacional, cuenta con soporte de varios idiomas y su diseño está orientado a interacciones seguras y responsables (Esden Business School, 2025). DeepSeek es también una herramienta conversacional de IA generativa que produce texto e imágenes, tiene la capacidad de mantener conversaciones en ingles, español o chino. Además de resolver problemas o muchas otras tareas que al usuario se le ocurrían. Es muy similar a GPT-4 con la diferencia que DeepSeek es de código abierto. lo que permite que cualquier usuario pueda entender como funciona, modificarlo y correrlo si así lo desea (Osornio, 2025).

Conociendo las capacidades de estos modelos de IA generativa para la búsqueda de información González Alcaide (2024) habla de como han evolucionado los procesos de búsqueda, en el pasado los operadores de búsqueda o terminología que identificaba un concepto era muy importante para la acceder a información, pero hoy en dia con la llegada de estos modelos de IA generativa, son otras habilidades que cobran protagonismo , como por ejemplo las de tipo lingüístico que relacionan

los elementos que integran las peticiones de información (Prompts), lo que da a entender que hoy en día puede ser mas importante saber como solicitar la información que conocer terminologías o conceptos.

Si bien son muchos los beneficios de la IA generativa, no todo es perfecto y existen algunos riesgos asociados al uso de estas herramientas. Por otra parte García-Peñalvo (2023) expone una lista de posibles riesgos de la inteligencia artificial generativa, entre las cuales se encuentran el impedir el correcto desarrollo de la creatividad, recibir respuestas incoherentes conocidas como alucinaciones, la no proporción de las fuentes o autores que respalden los resultados obtenidos, posible atribución de los resultados obtenidos , lo que podría considerarse como un plagio, entre otros riesgos, lo que da a entender que al final para maximizar sus beneficios es necesario un enfoque ético y responsable en su implementación.

2.1.1.5. Grandes modelo de lenguaje

Los grandes modelo de lenguaje (LLM) son modelos de aprendizaje profundo muy grandes que se entrenan previamente con enormes cantidades de datos. Su base es la arquitectura transformador, la cual utiliza codificadores y decodificadores con mecanismos de auto atención para interpretar secuencias de texto y comprender las relaciones entre palabras y frases.

A diferencia de las redes neuronales tradicionales como las redes neuronales recurrentes (RNN), los transformadores procesan datos en paralelo, lo que permite un entrenamiento más rápido utilizando GPUs. Esta arquitectura permite escalar los modelos a cientos de miles de millones de parámetros, utilizando datos masivos de fuentes como Internet, Common Crawl o Wikipedia.

Los LLM se destacan por su versatilidad, con características tales como: resumir documentos, traducir textos, responder preguntas y generar código. Aunque no son perfectos, han demostrado un rendimiento notable al hacer predicciones a partir de un numero relativamente pequeño de indicaciones o entradas (Amazon Web Services, 2024).

2.1.1.6. Modelos pre entrenados

Los modelos pre entrenados, como su nombre lo indica son modelos que han sido ajustados y entrenados con grandes volúmenes de datos, lo que al final permite a los desarrolladores ahorrar tiempo, la ventaja de estos modelos es su capacidad de ser reutilizados o ajustados para realizar nuevas tareas asignadas. Esto significa un ahorro considerable en tiempo y recursos que se gastarían en el entrenamiento inicial (Navarro, 2024).

No obstante, Navarro (2024) también da a conocer una lista de aplicaciones de IA en las que son útiles los modelos pre entrenados, entre los cuales se encuentran los siguientes:

- **Modelos para el procesamiento del lenguaje natural**
  Modelos conocidos como GPT-3 o BERT fueron entrenados con enormes cantidades de texto, lo favorece su implementación en chatbots, traducciones y generación de texto.

- **Modelos de visión por computadora**
  Modelos como ResNet o VGG son modelos de visión por computadora que se utilizan pa-

ra temas de reconocimiento facial o detecciones de objetos, esto es posible porque fueron previamente entrenados con millones de imágenes.

- **Modelos de redes generativas**
  Son un ejemplo de modelos pre entrenados que son útiles para la generación de imágenes, videos y otros tipos de contenidos.

2.1.1.7. Retrieval-Augmented Generation (RAG)

Según Zeichick (2023) la generación aumentada por recuperación (RAG) permite optimizar los resultados de un LLM (Large Language Model) sin necesidad de modificar el modelo que utilizan detrás, esto porque tiene la capacidad de recuperar información reciente y no estan limitados a los datos con los que los modelos subyacentes fueron entrenados. Gracias a esto puede generar respuestas utilizando datos recientes y enfocados en una organización o contexto en particular. En la **Figura 2.1** se observa un flujo básico del funcionamiento de la arquitectura RAG.

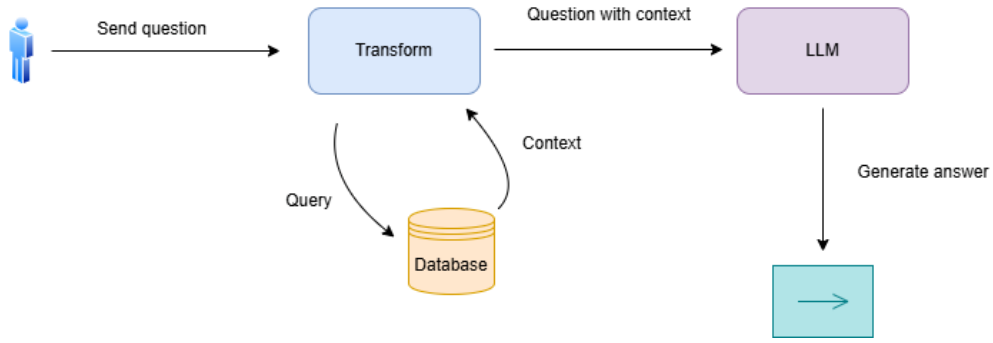


Figura 2.1: Diagrama arquitectura RAG
Nota: Figura de elaboración propia

Además las principales ventajas de la arquitectura RAG de acuerdo a Mollá (2024) son las siguientes:

- **Mejora en la precisión**
  Al utilizar fuentes de datos actualizadas, los LLM con RAG, pueden dar respuestas mas precisas y contextualizadas.

- **Reducción de alucinaciones**
  Los modelos de lenguaje tienden a dar respuestas que no tienen sentido o son incorrectas, RAG ayuda a reducir este fenómeno al poseer un contexto real y verificable.

- **Mejora del contexto**
  RAG permite a los modelos comprender mejor el contexto de las preguntas, lo que resulta en respuestas más coherentes y contextualizadas.

- **Reducción de costos**
  Al no entrenar un modelo desde cero o re entrenar uno constantemente con datos nuevos, RAG beneficia en la disminución de los costos asociados.

2.1.1.8. Bases de datos vectoriales

Las bases de datos vectoriales han surgido como solución para la recuperación de datos no estructurados, entre los cuales se encuentran texto, imágenes, audio video. Estas bases de datos se encargan de convertir estos datos en representaciones numéricas de alta dimensión usando técnicas de vector embeddings, esto permite realizar búsquedas basadas en similitudes. Su importancia ha ido en crecimiento debido al auge de la inteligencia artificial generativa y los grandes modelos del lenguaje (LLM) debido a la necesidad de manejo de grandes volúmenes de información no estructurada.

Una base de datos vectorial se define matemáticamente como un conjunto de tuplas de vectores, en donde cada uno de los valores está asociado a un identificador y a un conjunto de atributos. Funcionan basados en indexación y búsqueda de vectores a partir de consultas, utilizando métricas de distancia como la similaridad coseno y distancia euclidiana. Para optimar el rendimiento con grande volúmenes de datos, se utilizan técnicas de indexación como Inverted File (IVF) que consiste dividir los espacios vectoriales en una serie de clusters, donde cada cluster tiene información sobre vectores en forma de archivo invertido, IVF Flat, que es una combinación de la técnica anterior, usa (IVF) para dividir datos en clusters y utiliza el indice plano para búsquedas por fuerza bruta.

Gracias a su gran capacidad de gestionar datos complejos, las bases de datos vectoriales han servido para integrarse en sistemas de recomendación, análisis de documentos y procesamiento del lenguaje natural (Kukreja et al., 2023).

2.1.1.9. Arquitectura de software

Para Bass et al. (2021) la arquitectura de software es el conjunto de estructuras necesarias para razonar sobre un sistema, relacionando los elementos de software y las propiedades entre ellos.
De esa definición el termino elementos de software puede no ser claro, pero para comprenderlo mejor, es útil considerar los elementos como las partes de un software que se deben desarrollar, los cuales generalmente se conocen como módulos. Por otra parte, las propiedades hacen referencia a las interfaces, lo que síntesis significan la capacidad de los módulos de conectarse entre si, el correcto establecimiento de las interfaces es muy importante para la integración y el éxito de las pruebas de un sistema desarrollado por separado.

Es importante recalcar que el termino elementos de la definición de Bass et al. (2021) no hace referencia única y exclusivamente a módulos. El diseño de un sistema requiere pensar mas allá de los aspectos relacionados al desarrollo, sino también en satisfacer todos los requerimientos y la integración.

Además de identificar los módulos que separados van a construir el sistema, en la arquitectura de software es importante definir la manera en que se estructura un sistema ya que debe tener impacto en como satisfacer los requerimientos, en especial los que se consideran como atributos de

calidad, estos atributos son básicamente las características para establecer criterios sobre la calidad del sistema, no existe una lista definitiva de atributos para diseñar una arquitectura de software, pero entre los mas conocidos están la usabilidad, portabilidad, confiabilidad y seguridad.

La arquitectura de software tiene un ciclo de desarrollo, el cual tiene diferentes etapas que incluyen la identificación de atributos de calidad, alcance del proyecto, creación o refinamiento de la arquitectura, revisión de la arquitectura, decisión de llevar la arquitectura a producción o no, experimentación y la implementación. En la **Figura 2.2** se observa gráficamente este ciclo. Cada etapa es importante para asegurar que la arquitectura priorice los drivers arquitectónicos correctos, se alinee con los requerimientos y objetivos del sistema (Velasco-Elizondo, 2015).

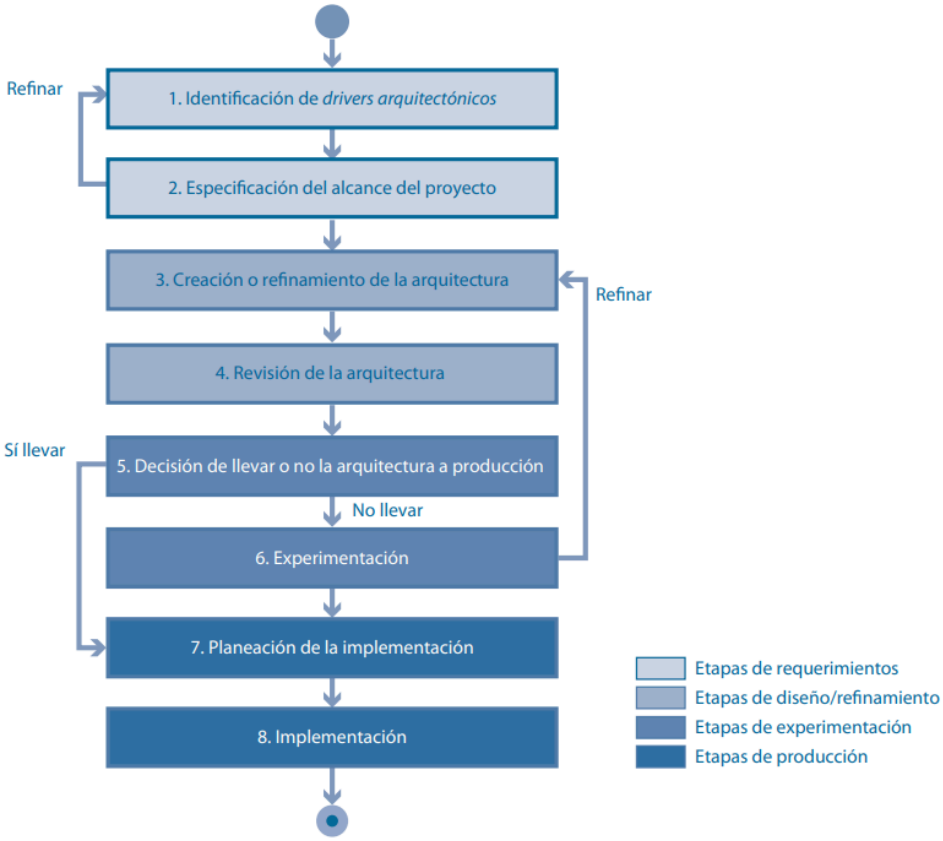


Figura 2.2: Ciclo de vida de la arquitectura de software

Nota: La figura representa el ciclo de vida de la arquitectura de software, diferenciando sus fases. *Fuente:* Perla Velasco-Elizondo, Arquitectura de Software: Conceptos y Ciclo de Desarrollo, Universidad Autónoma de Zacatecas, 2015.

Para concluir, se puede afirmar que la arquitectura de software es un proceso muy importante en el desarrollo de software, porque define la estructura que tendrá el sistema, una buena arquitectura de software garantiza la satisfacción de los requerimientos y atributos de calidad, garantizando la correcta funcionalidad del sistema y su mantenimiento a lo largo del tiempo.

2.1.1.10. Evaluación de la Recuperación en Sistemas RAG mediante RAGAS

RAGAS (*Retrieval-Augmented Generation Assessment*) es un marco de evaluación automatizada y sin referencias (*reference-free*) diseñado para medir la calidad de los sistemas *Retrieval-Augmented Generation* (RAG). En lugar de requerir respuestas de referencia o documentos anotados por humanos, RAGAS se apoya en modelos de lenguaje para realizar juicios automáticos sobre la calidad de la salida generada. Propone tres métricas principales: **Fidelidad**, **Relevancia de la Respuesta** y **Relevancia del Contexto** (Es et al., 2024).

1. Fidelidad (Faithfulness)

   La métrica de fidelidad evalúa si las afirmaciones presentes en la respuesta generada pueden ser justificadas por el contexto recuperado. Esta dimensión es importante para reducir alucinaciones y asegurar que la respuesta esté fundamentada.

   - Primero, se descompone la respuesta $as(q)$ en un conjunto de afirmaciones simples $S(as(q))$.
   - Luego, se verifica cada afirmación s_i respecto al contexto $c(q)$ mediante un modelo de lenguaje que indica si la afirmación es respaldada.

   La puntuación de fidelidad F se define como:

   $$F = \frac{|V|}{|S|}$$

   donde:

   - $|V|$ es el número de afirmaciones verificadas como correctas.
   - $|S|$ es el número total de afirmaciones extraídas de la respuesta.

2. Relevancia de la Respuesta (Answer Relevance)

   Esta métrica mide si la respuesta generada responde de manera directa y adecuada a la pregunta planteada, independientemente de su veracidad.

   - A partir de la respuesta $as(q)$, se generan n preguntas hipotéticas q_i.
   - Se calculan los *embeddings* de q y de cada q_i utilizando un modelo como `text-embedding-ada-002`.
   - Se computa la similitud coseno entre q y cada q_i.

   La puntuación de relevancia de la respuesta AR se define como:

   $$AR = \frac{1}{n} \sum_{i=1}^{n} \text{sim}(q, q_i)$$

   donde $\text{sim}(q, q_i)$ es la similitud coseno entre los *embeddings* de la pregunta original y cada pregunta generada.

3. Relevancia del Contexto (Context Relevance)

   La relevancia del contexto mide hasta qué punto el fragmento de texto recuperado contiene solo la información necesaria para responder la pregunta, penalizando la inclusión de contenido irrelevante o redundante.

   - El modelo de lenguaje extrae las oraciones relevantes desde el contexto $c(q)$ que son útiles para responder la pregunta q.
   - Si no se encuentra información suficiente, el modelo devuelve la frase "`Insufficient Information`".

   La puntuación de relevancia del contexto CR se define como:

$$CR = \frac{\text{número de oraciones extraídas}}{\text{número total de oraciones en } c(q)}$$

Las métricas anteriormente enumeradas fueron propuestas y validadas por Es et al. (2024) en el marco de evaluación RAGAS.

2.1.1.11. Bots en Telegram

Los bots en Telegram son cuentas especiales que no requieren un número de teléfono adicional para configurarse. Estas cuentas automatizadas se comunican con los servidores de Telegram mediante solicitudes HTTPS, lo cual permite a los desarrolladores gestionar mensajes entrantes y enviar respuestas programadas a los usuarios a través de la API de bots.

La creación de un bot comienza con el uso de BotFather, el bot oficial de Telegram para registrar nuevos bots y establecer sus configuraciones básicas, como nombre, descripción, comandos y avatar. Una vez creado, el bot puede interactuar en chats privados, grupos o canales, y responder automáticamente a eventos y mensajes.

Telegram permite dos métodos para conectar bots con los servidores: una consiste en que el bot consulte constantemente a los servidores de Telegram para saber si hay nuevos mensajes (lo que se conoce como long polling) y los webhooks, donde Telegram envía directamente las actualizaciones a una URL especificada por el desarrollador.

La plataforma también proporciona herramientas para crear interfaces de usuario enriquecidas, como teclados personalizados, botones inline y menús contextuales. Los bots pueden enviar y recibir diferentes tipos de contenido, incluyendo texto, imágenes, documentos, encuestas, stickers y más. Además, Telegram ofrece soporte para pagos integrados y comandos personalizados, lo que amplía las posibilidades de automatización e interacción dentro del ecosistema de mensajería.(Telegram, 2024)

2.1.1.12. DeepSeek en entornos locales

DeepSeek es un modelo de inteligencia artificial avanzado que puede ser auto alojado localmente, lo que permite un rendimiento más rápido, mayor confidencialidad en el manejo de los datos y

una configuración flexible adaptada a las necesidades del usuario. Aunque su versión completa requiere una infraestructura de alto rendimiento, sus variantes como DeepSeek-R1-Distill-Qwen-7B y DeepSeek-R1-Distill-LLaMA-70B, ofrecen una alternativa más eficiente y accesible para su implementación en entornos controlados (James, 2025).

Además, la posibilidad de desplegar DeepSeek en sistemas locales aporta un valor añadido significativo en términos de gobernanza de los datos. Esta capacidad permite a investigadores e instituciones ejecutar el modelo directamente en sus propios entornos, garantizando control absoluto sobre sus resultados y procesos (Wu, 2025).

2.2. Estado del Arte

El desarrollo de herramientas basadas en inteligencia artificial para la recuperación de información ha captado el interés de diversos desarrolladores e investigadores, quienes han abordado esta problemática desde distintos enfoques.

2.2.0.1. Demeter chatbot

Uno de estos enfoques es la creación de una herramienta de inteligencia artificial capaz de proporcionar información agroclimatica a productores y extencionistas de Colombia, Sotelo (2021) desarrolla Demeter chatbot, una herramienta de inteligencia artificial capaz de proveer información agroclimatica a productores, entre las cuales se encuentran pronósticos climáticos, fechas de siembra, cultivos disponibles, mejores fechas de siembra y rendimiento potencial de cultivos, esto se realizó utilizando procesamiento de lenguaje natural basados en BERT, esta herramienta es capaz de reconocer las intenciones de los usuarios, extraer las entidades relevantes como cultivares, localidades y fechas, y con esto buscar en una web Api la información para generar una respuesta.

La solución se divide en tres partes principales, que son demeter, melisas y la base de datos (Web Api), donde demeter es el núcleo del sistema, encargado de el análisis de mensajes y generación de respuestas, Melisa son los canales de comunicación (facebook y telegram), y la web Api que es donde se consulta la información dependiendo de las intenciones del usuario.

El sistema fue probado a través de telegram, logrando un aproximado de 473 interacciones, se evaluó la precisión de las respuestas y la utilidad de la información proporcionada. Los resultados arrojaron que demeter pudo responder adecuadamente en varios casos de uso logrando que un 44 % de los usuarios que probaron el sistema considerara útil la información que demeter les envió.

Si bien es un desarrollo robusto, no aborda por completo la problemática planteada, ya que su implementación no incluye ningún modelo de IA para generación de texto, ya que sus posibles respuestas están ya predefinidas desde un comienzo y solo se personalizan si se les concatena el resultado de la búsqueda. Esto limita la capacidad del sistema de generar respuestas personalizadas y naturales en función del contexto del usuario.

2.2.0.2. Chatbot para apoyo a victimas de acoso sexual

Uno de los desarrollos más recientes basados en LLM y algoritmos de recuperación de información es el de Vakayil et al. (2024). En este estudio desarrollaron un chatbot basado en Llama-2 utilizando la arquitectura de generación aumentada por recuperación (RAG), diseñado para dar apoyo a victimas de acoso sexual. El chatbot realiza extracciones de documentos legales en formato PDF, almacena información en ChromaDB y recupera la información utilizando LangChain, esto le permite generar respuestas empáticas, seguras y sin prejuicios, alanzando una precisión de hasta el 95 %, este desarrollo representa un avance importante en la creación de asistentes virtuales enfocados en temas sensibles, integra conceptos importantes de algoritmos de recuperación de información combinados con LLM, sin embargo no aborda del todo la problemática planteada puesto que es un sistema que corre unicamente en un entorno local, su falta de disponibilidad en diferentes canales de información puede limitar considerablemente su alcance a un publico objetivo.

2.2.0.3. Chatbot para tratamiento de mieloma múltiple

Otro de los enfoques recientes es un chatbot desarrollado por Quidwai and Lagana (2024) para asistir en el tratamiento de mieloma múltiple (MM), utilizando técnicas de generación aumentada por recuperación (RAG), este sistema permite a los médicos e investigadores acceder a información sobre el MM mediante métodos de búsqueda semántica, análisis de literatura y generación de respuesta usando IA. El modelo utilizado para este estudio fue Mistral-7B, optimizado para dar respuestas en el contexto clínico, la infraestructura de este sistema está alojado en Amazon kendra que ofrece una interfaz web personalizada con funciones de autenticación de usuario con amazon cognito. Para la recuperación de documentos se usa Amazon open search services, el cual permite la recuperación de documentos mediante incrustacions vectoriales. Además el sistema incorpora técnicas de clustering y análisis de datos exploratorios para mejorar la generación de respuestas.

Para evaluar los resultados del chatbot, se comparó con otras herramientas de IA, entre las cuales se encuentran GPT-3.5-turbo-16k y GPT-4-32k en un benchmark de preguntas clínicas acerca del MM diseñadas por oncólogos del Mount sinai hospital. Los resultados arrojaron que el modelo logro un rendimiento similar a GPT-4 en términos de recuperación de información medica, pero con un costo computacional mucho menor. Utiliza el modelo embebido de BAAI/bge-small-en-v1.5 que tiene una dimensión de 384, frente a los 1453 que usa GPT-4, esto optimiza su eficiencia sin llegar a comprometer la calidad de las respuestas y reduce el riesgo de alucinaciones.

Si bien este desarrollo es un avance significativo en la medicina de precisión, no aborda completamente la problemática planteada debido que su uso esta restringido a Amazon kendra, limitando su disponibilidad a otros especialistas. No esta integrado con ninguna red social lo que impide su uso por un publico más amplio.

2.2.0.4. Herramienta de monitoreo de puntos de agua en Etiopía

En Etiopía se implementó el sistema *Waterpoints Monitoring*, una herramienta digital desarrollada por la *Alianza de Bioversity International y el CIAT*, en colaboración con el *Ministerio*

de Agricultura de Etiopía y el *Instituto Etíope de Investigación Agrícola (EIAR)*. Esta plataforma forma parte del proyecto *Livestock Water Monitoring and Risk Management System (LWMRMS)*, financiado por la *Fundación Bill y Melinda Gates*.

El objetivo del sistema es brindar información sobre los recursos hídricos en regiones pastoriles, mejorando la preparación de las comunidades frente a condiciones climáticas impredecibles. Ofrece datos integrados sobre niveles de agua superficial, condiciones de pastoreo y pronósticos climáticos de mediano plazo, utilizando sensores en campo, imágenes satelitales y modelos hidrológicos calibrados. Estos modelos se adaptan específicamente a las condiciones del flujo de agua superficial en zonas áridas y semiáridas, donde la lluvia es estacional y los puntos de agua son temporales pero críticos para la ganadería.

Con esto, los usuarios pueden acceder a información específica de cada punto de agua como la profundidad del agua, los niveles de precipitación y la tasa de evaporación superficial. Esta información resulta vital para los pastores que gestionan rebaños con necesidades de agua cambiantes y deben planificar sus movimientos en función de la disponibilidad de recursos.

La plataforma permite visualizar los cambios en el almacenamiento de agua en puntos importantes, estimar escenarios futuros con hasta seis meses de antelación, y calcular la cantidad de días necesarios para que el ganado alcance dichos puntos. Esta capacidad mejora significativamente la toma de decisiones en cuanto al movimiento y manejo de los rebaños, reduciendo los riesgos asociados a la escasez de agua y forraje(Alemayehu et al., 2024b).

Aunque esta solución aborda el acceso a información actualizada sobre puntos de agua de forma visual y técnica, dicha información aún no está disponible mediante canales conversacionales o interfaces interactivas.

En la **Tabla 2.1** se muestra una comparación de los estudios analizados, destacando el contexto geográfico que abordan y las brechas que buscan cubrir en sus respectivos dominios de aplicación.

Estudio / herramienta	**Dominio**	**Uso de LLM / RAG**	**Interfaz**	**Contexto geográfico**	**Brecha cubierta por este trabajo**
Demeter chatbot, Sotelo (2021).	Agricultura	No usa LLM ni RAG	Telegram	Colombia	Facilita el acceso a información agroclimatica a través de preguntas frecuentes en Telegram, útil para pequeños productores sin conexión permanente, limitado a intenciones predefinidas, sin generación de información.
Chatbot para apoyo a victimas de acoso sexual, Vakayil et al. (2024).	Género / Legal / Derechos humanos	Usa LLM y RAG	Entorno local	India	Apoya a víctimas de violencia sexual, combinando lenguaje natural con recuperación de información legal en un entorno protegido, restringido su uso a un entorno local.
Chatbot para tratamiento de mieloma múltiple, Quidwai and Lagana (2024).	Salud / Oncología	Usa LLM y RAG	Amazon Kendra	New York, NY, USA	Brinda información especializada en oncología para pacientes y médicos, integrando respuestas generadas desde documentos clínicos con trazabilidad usando RAG, restringido a amazon kendra.
Herramienta de monitoreo de puntos de agua en Etiopía, Alemayehu et al. (2024b),	Agroclimático	No usa LLM ni RAG	Web / Móvil	Etiopía	Provee datos periódicos sobre estado y disponibilidad de puntos de agua para agropastores en Etiopía, mediante mapas y visualizaciones interactivas, no aplica modelos LLM.
Propuesta de este trabajo	Agroclimático	**LLM + RAG**	Telegram (caso posible de uso)	Etiopía	Integra arquitectura RAG con LLM, acceso por lenguaje natural y despliegue en interfaz conversacional, replicable en otros países del Cuerno de África

Tabla 2.1: Comparación entre trabajos previos y la propuesta actual

2.3. Resumen del capítulo

En este capítulo se presentaron las bases teóricas y los trabajos previos relacionados con el proyecto. En primer lugar, se abordaron los conceptos necesarios para contextualizar el proyecto, incluyendo los sistemas de recuperación aumentada con generación (RAG), LLMs, los principios de la arquitectura de software, y el uso de bases de datos vectoriales para la representación y recuperación semántica de información.

Posteriormente, se abordaron diferentes trabajos relacionados con el proyecto, permitiendo identificar enfoques existentes, técnicas empleadas por otros autores y resultados previamente reportados en la literatura. Este análisis comparativo permitió establecer similitudes y diferencias entre los trabajos existentes y la solución planteada en este proyecto.

Finalmente, se contrastaron las fortalezas y limitaciones de los trabajos revisados frente a la propuesta desarrollada, evidenciando los aportes específicos de este trabajo. De esta manera, el capítulo proporcionó el marco conceptual y el sustento teórico necesario para el desarrollo e implementación de la herramienta presentada en los capítulos posteriores.

Capítulo 3

Desarrollo del Proyecto

Este capítulo presenta el desarrollo de una herramienta que, mediante el uso de lenguaje natural, modelos de lenguaje de gran tamaño (LLM) y una arquitectura de Recuperación Aumentada por Generación (RAG), permite responder consultas relacionadas con la disponibilidad de puntos de agua en Etiopía a través de un canal conversacional implementado en Telegram.

En este capítulo se abordan los aspectos necesarios para el desarrollo exitoso de la herramienta. Se describe el uso de los datos existentes de puntos de agua, la definición de qué información será vectorizada y con qué periodicidad, el diseño de la arquitectura del sistema, el proceso de desarrollo de la herramienta.

3.1. Vectorización, acceso, almacenamiento y actualización de datos de puntos de agua

3.1.1. Vectorización de datos de puntos de agua

Para la vectorización de los datos de los puntos de agua se debe tener en cuenta que los registros que conforman el núcleo de la herramienta corresponden a información climática. Estos registros incluyen variables como evapotranspiración, lluvia y profundidad de cada punto de agua, registradas diariamente para cada punto durante un período superior a 20 años.

Actualmente, estos datos se encuentran almacenados en una base de datos MongoDB con la siguiente estructura:

```
{
  "_id": "ObjectId('64d1bf35972bbad10c0c8590')",
  "date": "2008-03-09T00:00:00.000+00:00",
  "values": [
    { "type": "depth", "value": 0 },
    { "type": "evp", "value": 5.18436 },
    { "type": "rain", "value": 0 },
    { "type": "scaled_depth", "value": 0 }
  ],
  "waterpoint": "ObjectId('64d1bf1fcc703fe54e05ee7d')"
}
```

En conjunto, esta información representa la mayor parte de la base de datos, con más de 20 años de registros de 41 puntos de agua distintos, lo que equivale a más de 370.000 documentos unicos. Su estructura y tipos de datos son estables y no se espera que cambien con el tiempo.

Aunque era técnicamente viable vectorizar estos datos, su alto volumen, su formato estrictamente estructurado y la posibilidad de consultarlos de manera directa mediante consultas precisas en MongoDB hacen innecesaria su vectorización. Además, este proceso habría implicado la duplicación del tamaño de la base de datos, al generar una base vectorial adicional que contendría los mismos registros representados como embeddings.

En consecuencia, estos datos se mantuvieron en su formato original y se accedió a ellos mediante consultas directas, esto con el fin de reducir el usod de recursos computacionales como el almacenamiento del sistema.

Por otro lado, existen datos asociados a los puntos de agua que presentan un conjunto de información más descriptiva y variable en el tiempo. Estos datos pueden ser agregados utilizando contenidos que evolucionan a lo largo del tiempo, e incluyen texto plano incorporado por administradores del sistema.

Este tipo de información está asociada directamente a cada punto de agua y puede contener datos como la historia del punto de agua, su año de construcción, el número de personas que lo utilizan, la cantidad de cabezas de ganado en los alrededores, así como cualquier otro dato que se considere importante agregar, Un ejemplo de este tipo de datos se presenta a continuación.


```
{
  "_id": "64e66483c3b3f6f36b810e7c",
  "content": {
    "title": "general",
    "type": "text",
    "values": [
      {
        "content": "Before filled with silt, the total population of 7,859
            individuals relied on the waterpoint for drinking..."
      },
      {
        "waterpoint": "ObjectId('64d1bf1fcc703fe54e05ee7d')"
      }
    ]
  }
}
```


Se observa que, los datos climáticos, los cuales presentan variables bien definidas como fechas y valores específicos que pueden ser utilizadas para ejecutar consultas precisas en la base de datos MongoDB, en este tipo de información la situación es distinta. Estos datos corresponden a textos que pueden alcanzar una longitud considerable, lo que dificulta el acceso directo a una sección específica cuando el usuario realiza una consulta concreta.

Por consiguiente, y con el fin de facilitar la recuperación de información mediante lenguaje natural, se decidió vectorizar estos datos y representarlos mediante embeddings, permitiendo así

realizar búsquedas semánticas dentro de la arquitectura RAG.

Teniendo en cuenta que ya se contaba con una base de contenido almacenada en la base de datos, fue necesario vectorizar dicha información en una primera instancia, con el fin de disponer de una base vectorial que permitiera probar y validar el funcionamiento de la herramienta.

Para ello, se desarrolló un algoritmo en Python que se conectaba a la base de datos MongoDB, extraía el contenido asociado a cada punto de agua, generaba sus representaciones vectoriales mediante embeddings y almacenaba tanto los vectores resultantes como su metadata correspondiente en la base de datos vectorial seleccionada.

Durante las pruebas de la herramienta surgió un reto importante que fue necesario abordar. Se identificó que muchos contenidos presentaban alta similitud entre diferentes puntos de agua, lo que en algunas ocasiones provocaba la recuperación de textos asociados a puntos distintos al que se hacía referencia en la consulta.

Para solucionar este problema, durante el proceso de vectorización se implementó una estrategia adicional. Antes de generar los embeddings a partir del texto, se accedía a la colección donde se almacenan los puntos de agua y se extraía el nombre correspondiente a cada uno. Dicho nombre se incorporaba al inicio del contenido bajo el formato "Waterpoint: nombre del punto de agua", asegurando que cada texto quedara explícitamente asociado a su respectivo punto de agua.

Nota: El script utilizado para el proceso de importación de datos vectorizados puede ser consultado en el siguiente enlace: Script de importación de datos.

Con esta estrategia se logró mejorar la diferenciación entre contenidos similares, ya que, incluso en presencia de textos con estructuras o descripciones parecidas, el nombre del punto de agua actúa como un elemento distintivo dentro del embedding. No obstante, este enfoque puede presentar limitaciones en escenarios donde múltiples puntos de agua comparten nombres muy similares, lo cual se considera un posible riesgo a tener en cuenta en trabajos futuros. A continuación, se muestra un documento real antes y después de ser vectorizado.

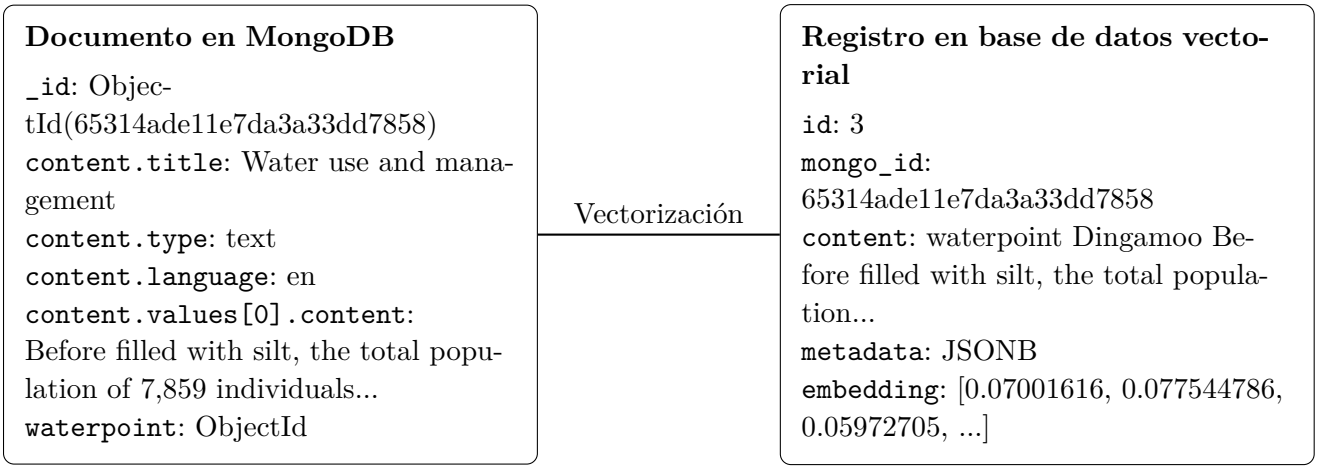


Figura 3.1: Representación conceptual de la transformación de un documento desde MongoDB hacia su representación en la base de datos vectorial

3.1.2. Acceso a los datos

El acceso a los datos se implementó siguiendo un enfoque de doble vía, lo que permitió diferenciar de manera explícita cuándo se debía acceder a los datos climáticos almacenados en MongoDB y cuándo a los datos descriptivos almacenados en la base de datos vectorial. Esta diferenciación se realizó a través de la detección de la intención de la consulta del usuario.

Cuando la consulta estaba orientada a datos climáticos, se extraían de la pregunta variables específicas y referencias temporales, como fechas, las cuales se utilizaban para construir consultas directas sobre la base de datos MongoDB. Por el contrario, cuando la consulta estaba enfocada en información descriptiva, el sistema recurría a la base de datos vectorial para realizar una búsqueda semántica sobre los contenidos previamente vectorizados.

En resumen, la herramienta identifica la intención del usuario y, dependiendo de dicha intención, accede a una u otra fuente de datos. Para el acceso a la base de datos MongoDB se emplearon consultas estructuradas utilizando la ORM MongoEngine, mientras que, para los datos vectorizados, el sistema recurría a una base de datos vectorial PostgreSQL con la extensión pgvector, accedida mediante el conector psycopg2, lo que permitía ejecutar consultas de similitud basadas en embeddings.

Nota: El proceso de detección de la intención del usuario se describe con mayor detalle en secciones posteriores.

3.1.3. Actualización de los datos

Después de vectorizar los datos ya existentes, es necesario definir un enfoque que permita mantener actualizadas tanto la base de datos vectorial como la base de datos climática a lo largo

del tiempo, con el fin de que la herramienta conserve la capacidad de responder a las consultas utilizando la información más actualizada posible. Para ello, el proceso se dividió en dos enfoques diferenciados, uno orientado a la actualización de los datos vectorizados y otro a la actualización de los datos climáticos, los cuales se describen a continuación.

1. **Datos climaticos** Para la actualización de los datos climáticos, cabe resaltar que todo este proceso es orquestado por el *Ethiopian Institute of Agricultural Research (EIAR)*, entidad encargada de enviar los datos climáticos al *Centro Internacional de Agricultura Tropical (CIAT)* mediante el uso de una *API RESTful*. Para esto, el sistema mantiene una variable en la que se almacena la fecha de la última actualización de datos enviada a CIAT.

   De manera diaria, se ejecuta un proceso automatizado que valida si los datos climáticos disponibles en EIAR se encuentran adelantados en el tiempo, es decir, si existen nuevos registros que aún no han sido enviados a CIAT. En caso de identificarse información nueva, el sistema activa automáticamente un proceso de envío que transmite los datos climáticos comprendidos entre la última fecha de actualización registrada y la fecha más reciente disponible en EIAR. Una vez recibidos, estos nuevos registros son almacenados en la base de datos *MongoDB*, actualizando así la información disponible para la herramienta. Finalmente, la variable que almacena la fecha de la última actualización se actualiza con la fecha más reciente enviada, lo que garantiza la sincronización continua de los datos.

   En la **Figura 3.2** se muestra el proceso descrito.

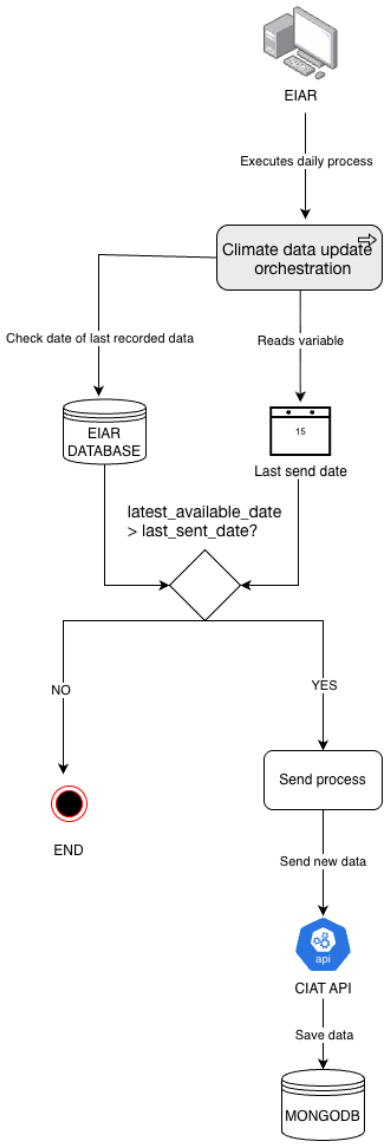


Figura 3.2: Flujo de actualización automática de datos climáticos y de puntos de agua
Nota: Figura de elaboración propia

2. **Datos descriptivos** Para este proceso, primero se debe resaltar que ya existe una herramienta en operación encargada de la actualización de la información descriptiva asociada a los puntos de agua, la cual almacena estos datos en la base de datos *MongoDB*. Esta herramienta fue desarrollada en el framework *Flask* y está orientada a los administradores del

proyecto, quienes pueden agregar, editar o eliminar contenido relacionado con un punto de agua específico mediante una interfaz gráfica.

Cada una de estas acciones se refleja en la base de datos MongoDB, garantizando la persistencia y actualización del contenido original. Sobre este flujo ya existente, se incorporó un proceso adicional para la vectorización de los datos. De forma paralela al almacenamiento del contenido en MongoDB, se ejecuta un proceso que se encarga de vectorizar el contenido textual recién agregado o modificado mediante un *modelo de embeddings*, y de almacenarlo en la base de datos vectorial correspondiente.

Con este enfoque, se asegura que la información descriptiva vectorizada se mantenga continuamente actualizada y sincronizada con respecto a los cambios realizados por los administradores del sistema.

En la **Figura 3.3** se muestra el proceso descrito.

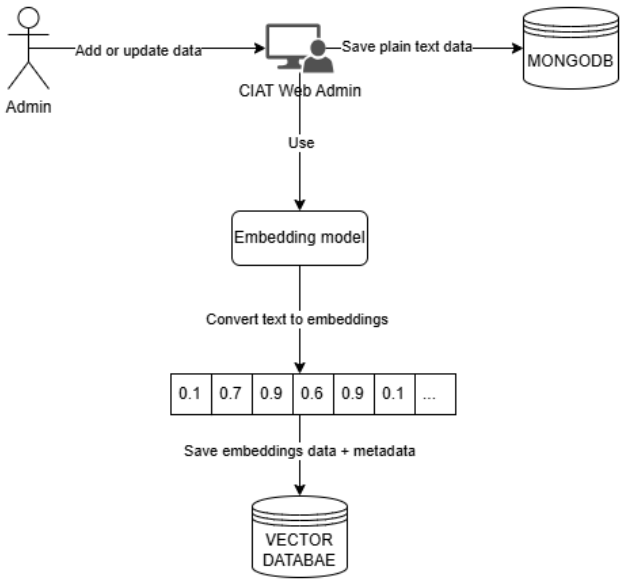


Figura 3.3: Flujo de actualización automática de datos descriptivos de puntos de agua
Nota: Figura de elaboración propia

Con este enfoque, la herramienta mantiene sincronizados tanto los datos exactos obtenidos de mediciones diarias como la información descriptiva, asegurando poder brindar información actualizada para cada consulta.

3.2. Arquitectura del sistema

Antes de abordar en detalle el desarrollo de la arquitectura de software, es importante aclarar que durante la fase inicial del desarrollo del proyecto se exploraron diversas alternativas de arquitectura para la implementación del sistema.

Entre las opciones consideradas se encuentran plataformas de orquestación de flujos como *n8n*, que es una plataforma que permite conectar aplicaciones y datos con nodos sin necesidad de programar e incluye también flujos de inteligencia artificial (Sofia Hansen, 2025), así como frameworks especializados en aplicaciones basadas en modelos de lenguaje, como *LangChain*, que es un marco de trabajo de código abierto para crear aplicaciones basadas en modelos de lenguaje de gran tamaño (LLM), que puede integrarse con herramientas de Amazon web services (AWS) como Amazon Bedrock, Amazon Kendra, Amazon SageMaker JumpStart, con LangChain los desarrolladores pueden interactuar con modelos como GPT, Bard y PaLM, ademas de que permite diseñar sistemas RAG con bastantes herramientas para buscar y almacenar información que refina las respuestas de los modelos de lenguaje (Amazon Web Services, 2025).

No obstante, el propósito de este sistema es su entrega y control por parte del *EIAR*, lo cual requiere una solución que pueda ser desplegada, mantenida y modificada de manera autónoma. Por esta razón, se trato de evitar depender de plataformas externas o servicios de terceros que impliquen costos recurrentes o limiten el control sobre la infraestructura. Dado que los recursos del proyecto son limitados, se priorizó el uso de tecnologías que pudieran ejecutarse y administrarse localmente.

Por esta razón, se optó por desarrollar una arquitectura basada en código, modular y desacoplada, que permite un control completo sobre los flujos de información, los mecanismos de validación y los componentes internos.

Otro aspecto importante a aclarar es que, si bien inicialmente se planteó el uso exclusivo de modelos de lenguaje locales, durante el desarrollo del sistema se presentaron limitaciones técnicas relacionadas con el uso de memoria RAM. Al ejecutar todo el proceso utilizando únicamente modelos locales, el consumo de memoria aumentaba de forma abrupta hasta alcanzar el 100 % de la capacidad disponible del entorno de desarrollo, lo que provocaba que el sistema no pudiera generar respuestas y colapsara por falta de recursos.

Ante esta situación, se optó por adoptar un enfoque híbrido que combina el uso de un modelo de lenguaje local y un modelo de lenguaje en la nube. El modelo local se emplea para tareas de menor complejidad, como la detección de la intención del usuario y la validación del dominio de la consulta, mientras que el modelo en la nube se utiliza para tareas más demandantes, como la generación de respuestas, la extracción de variables y el análisis del contexto recuperado.

Este enfoque permite delegar las operaciones más costosas en términos de recursos a un servicio

externo, reduciendo el consumo de memoria local y garantizando el funcionamiento estable del sistema. Adicionalmente, se decidió utilizar un modelo de embeddings basado en la nube, dado su bajo costo por token. Si bien hubiera sido posible implementar todo el sistema utilizando exclusivamente modelos en la nube, se optó por una arquitectura híbrida con el fin de equilibrar el consumo de recursos locales, los costos operativos y el control sobre la infraestructura.

Finalmente, se aclara que inicialmente se había planteado el uso del modelo *DeepSeek* como modelo de lenguaje local. No obstante, durante las pruebas de implementación se comparó su desempeño con el modelo *Llama 3.1*, observándose que este último presentaba un menor consumo de memoria RAM por ejecución en el entorno de desarrollo utilizado.

Dado que la funcionalidad del sistema estaba directamente condicionada por el uso de recursos de memoria, se decidió adoptar *Llama 3.1* como modelo local, al permitir una ejecución que consume menos recursos sin comprometer el rendimiento esperado. Llama 3.1 es un modelo de lenguaje de código abierto desarrollado por Meta con una ventana contextual de 128K, el cual a diferencia de varios modelos comerciales, se puede descargar y personalizar segun las necesidades de los desarrolladores (Meta, 2024).

El sistema fue diseñado para que usuarios agropastores, extensionistas, tomadores de decisiones en Etiopía y usuarios en general puedan acceder a información proveniente de diferentes puntos de agua ubicados en el país mediante consultas en lenguaje natural, una interfaz conversacional, en este caso, la aplicación de Telegram. La arquitectura del sistema fue elaborada utilizando el modelo **C4**, el cual permite dividir el diseño en tres niveles:

1. **Contexto del sistema:** muestra cómo el sistema se relaciona con los actores externos y los sistemas con los que interactúa.
2. **Contenedores del sistema:** describe la organización interna en términos de aplicaciones, bases de datos y servicios principales.
3. **Componentes de cada contenedor:** detalla las funcionalidades específicas de cada uno de los contenedores y cómo se interconectan para dar soporte al sistema en su conjunto.

3.2.1. Nivel 1

El sistema está compuesto por cuatro actores principales: **Water Bot**, **Core system**, **LLM Model** y **Embedding model**, así como por una entidad externa denominada **Conversational interfaces**, que representa los distintos tipos de interfaces conversacionales a través de los cuales los usuarios pueden interactuar con el sistema. A continuación, se describe cada uno de estos actores y entidades.

Conversational Interfaces representa el conjunto de canales a través de los cuales los usuarios pueden interactuar con el sistema, tales como aplicaciones web, plataformas de mensajería u otros

medios conversacionales que puedan requerirse en el futuro. Esta capa es responsable de recibir las consultas del usuario, transmitirlas al **Water Bot**.

Water Bot es el agente enfocado en el dominio de recursos los puntos de agua y su información de clima. Su función principal es recibir las consultas recibidas desde las interfaces conversacionales, transformarlas en un formato estructurado y enviarlas al **Core system** para su procesamiento.

Adicionalmente, Water Bot recibe la respuesta generada por el **Core system** y la transmite de vuelta a las interfaces conversacionales para que sea entregada al usuario.

Core system es el componente central de procesamiento y orquestación del sistema. Su función principal es analizar la intención del usuario a partir de la consulta estructurada recibida desde el **Water Bot**, con el fin de determinar si la solicitud corresponde a información climática, variables climáticas específicas, información general sobre un punto de agua o el estado actual de un punto de agua.

Asimismo, el **Core system** evalúa si la consulta se encuentra dentro del dominio del sistema, con el propósito de establecer si puede ser respondida o no. Adicionalmente, se encarga de validar la consulta para detectar solicitudes potencialmente dañinas o que requieran información sensible o peligrosa.

Adicionalmente, el **Core system** gestiona el historial de las conversaciones por usuario, almacenando y recuperando el contexto y las respuestas generadas previamente, con el fin de evitar consultas innecesarias a las bases de datos cuando no son estrictamente requeridas.

Finalmente, el **Core system** es responsable de construir las consultas necesarias para la recuperación de información, tanto desde la base de datos de MongoDB como desde la base de datos vectorial utilizada para la búsqueda semántica, integrando los resultados como contexto para la generación de la respuesta.

LLM Model corresponde al modelo de lenguaje utilizado como componente de apoyo para la generación de respuestas a partir de las consultas del usuario. Su función principal es asistir al **Core system** en la validación de la intención del usuario, la verificación de si la consulta se encuentra dentro del dominio del sistema, la detección de solicitudes potencialmente peligrosas o inapropiadas, y la generación de la respuesta en lenguaje natural a partir del contexto proporcionado.

De esta manera, el **LLM Model** no actúa como el núcleo de la lógica del sistema, sino como un servicio externo que aporta capacidades de comprensión del lenguaje natural para fortalecer los procesos de clasificación, validación y control de las consultas.

Embedding Model es el encargado de transformar las consultas del usuario en representaciones vectoriales (embeddings), las cuales son utilizadas para realizar búsquedas por similitud en la base de datos vectorial. Este proceso permite recuperar información semánticamente relacionada con la consulta, que posteriormente es utilizada por el **Core system** como contexto para la generación de la respuesta.

En la **Figura 3.4** se puede observar la representación gráfica del nivel 1.

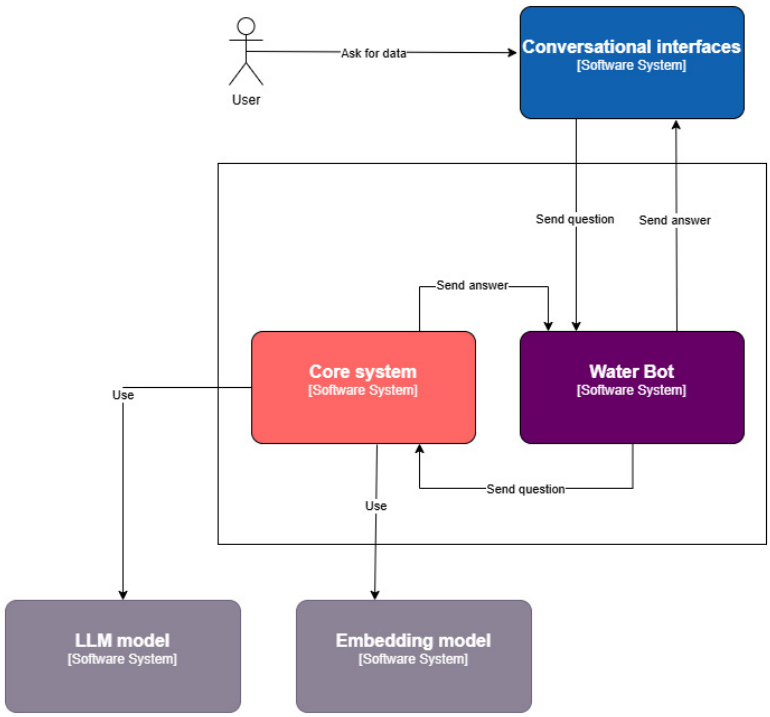


Figura 3.4: Nivel 1, Diagrama de contexto del sistema
Nota: Figura de elaboración propia

3.2.2. Nivel 2

En el **Nivel 2** se describen todos los contenedores que conforman la solución. El sistema cuenta con un total de cinco contenedores: tres contenedores para las bases de datos, un contenedor para el *Water Bot* y un contenedor para el *Core System*.

A diferencia del **Nivel 1**, en este nivel la entidad **Conversational interfaces** se representa como una única instancia denominada *Telegram App*, dado que esta aplicación de mensajería es el canal de interacción seleccionado para la implementación y validación del sistema en el presente trabajo.

Los contenedores correspondientes al *Core System* y al *Water Bot* fueron desarrollados en *Python 3.12*. En cuanto a las bases de datos, el sistema utiliza *MongoDB 7.0* como base de datos transaccional, *PostgreSQL 16* extendida con *pgvector* para la gestión de datos vectoriales, y *Redis 7* para el almacenamiento del historial conversacional.

En la **Figura 3.5** se puede observar la representación gráfica del nivel 2.

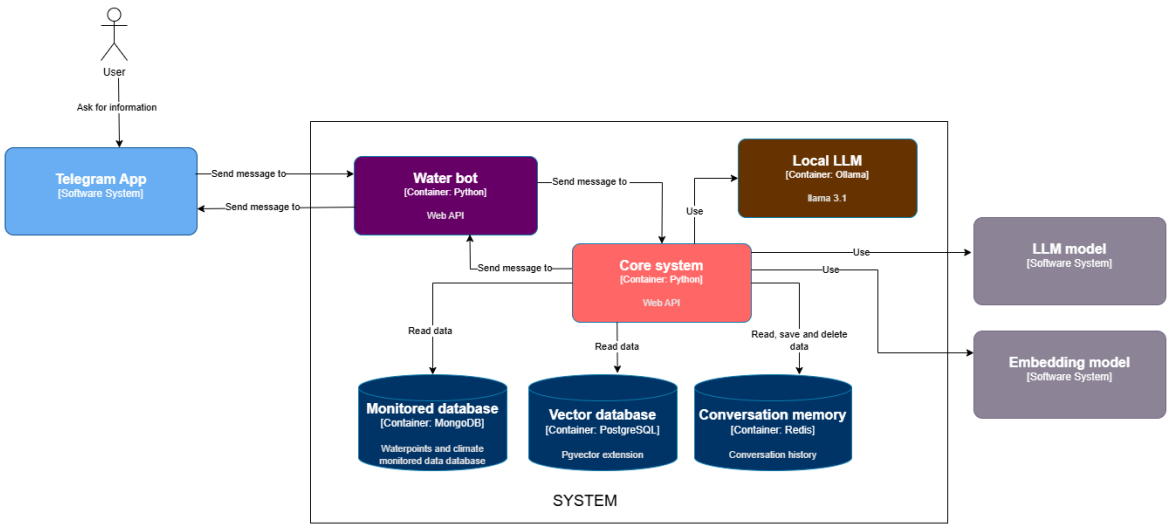


Figura 3.5: Nivel 2, Diagrama de contenedores del sistema
Nota: Figura de elaboración propia

Water Bot es un servicio backend que actúa como puente entre la aplicación de mensajería y el sistema central. Expone un endpoint de tipo *webhook* para recibir los eventos enviados por Telegram, normaliza los mensajes entrantes a un formato estructurado, los envía al **Core system** para su procesamiento y, posteriormente, envia la respuesta generada de vuelta al usuario a través de Telegram.

Core system es el componente central de procesamiento y orquestación del sistema. Recibe las consultas estructuradas desde el **Water Bot**, analiza la intención del usuario, valida que la solicitud se encuentre dentro del dominio de la aplicación y detecta consultas potencialmente inapropiadas o peligrosas.

Adicionalmente, el **Core system** construye y ejecuta las consultas necesarias para la recuperación de información desde las dos fuentes de datos, la base de datos de MongoDB y la base de datos pgvector, integrando los resultados como contexto para la generación de la respuesta.

Finalmente, el **Core system** coordina la interacción con el modelo de lenguaje para la generación de la respuesta en lenguaje natural y devuelve el resultado al **Water Bot** para su entrega al usuario.

Local LLM es el contenedor que aloja el modelo de lenguaje utilizado para el análisis de las consultas del usuario y la gestión del dominio conversacional. Su función principal es evaluar si una consulta se encuentra dentro del dominio del sistema y determinar el flujo de procesamiento correspondiente.

En particular, el **Local LLM** clasifica cada solicitud como válida o no válida para el sistema,

retornando las etiquetas *ok*, *out_of_scope* o *inappropriate*, las cuales permiten al **Core system** decidir si continuar con el procesamiento normal de la consulta o generar una respuesta alternativa al usuario.

Adicionalmente, el **Local LLM** detecta la intención del usuario, clasificando las consultas en categorías predefinidas como *CLIMATE_INFO* (información climática), *WATERPOINT_STATUS* (estado del punto de agua) y *GENERAL_INFO* (información general del punto de agua), las cuales son utilizadas por el **Core system** para enrutar la consulta y seleccionar la base de datos que se debe consultar.

Nota: *Cuando se hace referencia al estado de un punto de agua, se alude a la aplicación de una regla de negocio definida dentro del sistema. Dicha regla se basa en una fórmula que toma como referencia la última medición disponible de profundidad del punto de agua. Este valor es comparado con un conjunto de umbrales previamente establecidos y, en función del rango en el que se ubica la profundidad medida, se determina el estado actual del punto de agua.*

Monitored database es la base de datos encargada de almacenar la información estructurada relacionada con los puntos de agua y sus variables asociadas, tales como nombre del punto de agua y variables de clima (lluvia, profundidad, evapotranspiración).

Esta base de datos es utilizada por el **Core system** como fuente de información para responder a las consultas del usuario, permitiendo recuperar los datos necesarios sobre el estado actual y variables climáticas,

Vector database está implementada en *PostgreSQL* con la extensión *pgvector* la cual es una extensión de código abierto para PostgreSQL la cual permite almacenar y buscar en incrustaciones (embeddings) generadas por aprendizaje automático y es conocido por su robustez y capacidad para gestionar grandes conjuntos de datos (Avthar Sewrathan, 2023). En esta base de datos se almacenan representaciones vectoriales de información asociada a los puntos de agua, incluyendo descripciones generales, datos demográficos, de genero y otros atributos relacionados. Gracias a este esquema de almacenamiento, es posible realizar búsquedas semánticas y de similitud. El *Core System* utiliza esta información para enriquecer el contexto enviado al *LLM*, lo que facilita la generación de respuestas.

LLM model es un servicio de modelo LLM externo invocado una vez que la consulta ha sido validada como apropiada y dentro del dominio del sistema. A partir del texto de la consulta del usuario y de la lista de puntos de agua disponibles obtenida desde la base de datos, el servicio identifica el punto de agua al que hace referencia el usuario, incluso cuando el nombre es ingresado de forma incompleta, con variaciones ortográficas o con errores tipográficos, para este trabajo se deicidio trabajar con el modelo GPT-4o-mini debido a su bajo costo por tokens, siendo este de 15 centavos por millón de tokens de entrada y 60 centavos por millón de tokens de salida. (OpenAI, 2024a)

Adicionalmente, el servicio extrae las variables climáticas o atributos específicos solicitados (por ejemplo, lluvia, profundidad o evapotranspiración) y, utilizando el contexto proporcionado por el

Core system, genera la respuesta en lenguaje natural que será devuelta al usuario.

Embedding model es el modelo externo encargado de transformar las consultas del usuario en representaciones vectoriales (embeddings), las cuales son utilizadas para realizar búsquedas por similitud en la **Vector database**, para este trabajo se decidio trabajar con el modelo text-embedding-3-small ya que tiene un costo por token muy bajo, de 0.02 dolares por millon de tokens. (OpenAI, 2024b)

Su función principal consiste en recibir una consulta en formato de texto y convertirla en un vector numérico que permita comparar semánticamente dicha consulta con los documentos previamente almacenados en la base de datos vectorial. De esta manera, se posibilita la recuperación de información relevante aun cuando no exista una coincidencia exacta entre los términos utilizados por el usuario y los documentos almacenados.

Adicionalmente, este mismo modelo fue utilizado en la etapa de actualización y vectorización de los datos.

3.2.3. Nivel 3: Componentes del sistema

En el nivel 2 se describieron los contenedores principales que conforman la arquitectura del sistema: *Water Bot*, *Core System*, *Local LLM*, las bases de datos *Monitored database, Vector database y Conversation history* las dos dependencias externas *LLM model y Embedding model*. En este nivel se detalla la estructura interna de cada uno de estos contenedores, presentando los componentes que los conforman y las interacciones que se establecen entre ellos. El diagrama de la **Figura 3.6** muestra la disposición general de los componentes y el flujo de información entre los diferentes módulos del sistema.

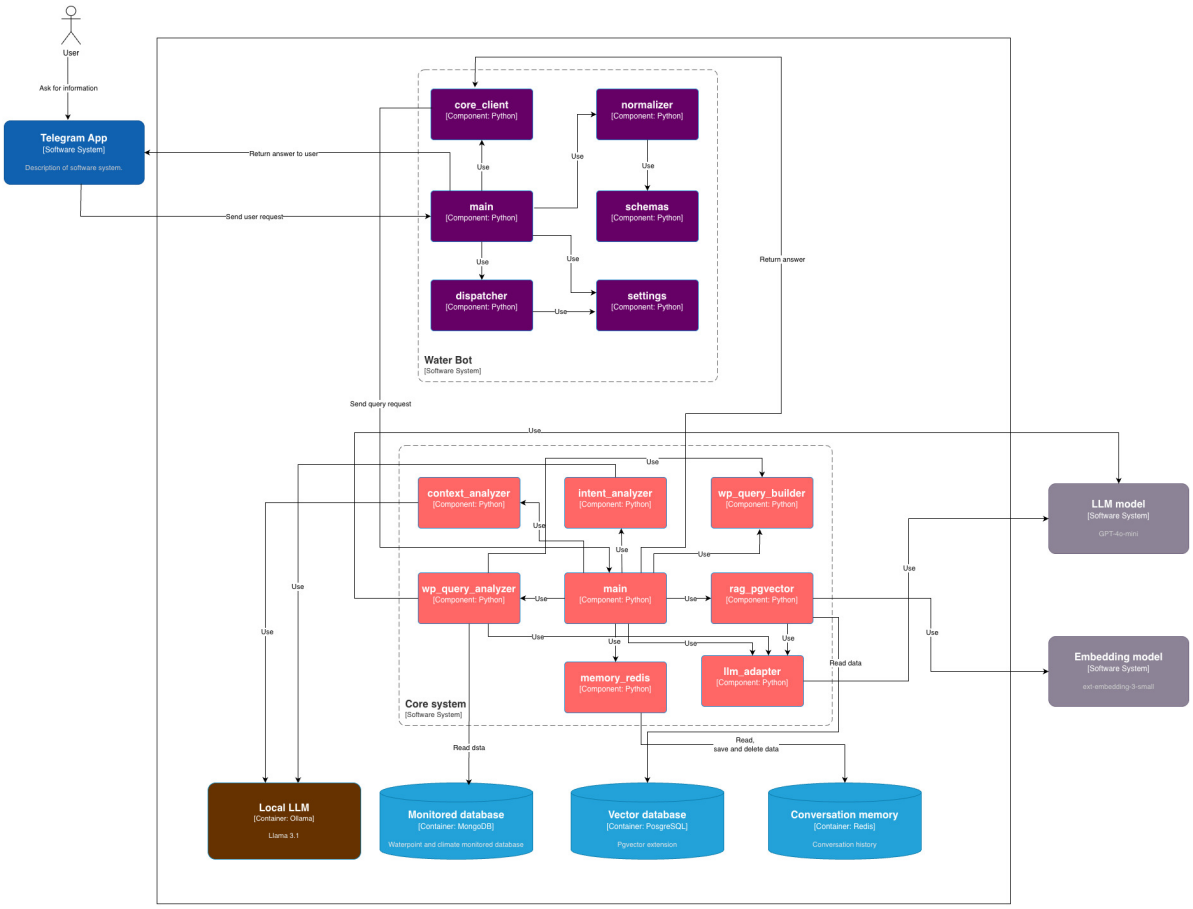


Figura 3.6: Nivel 3, Diagrama de componentes internos del sistema
Nota: Figura de elaboración propia

Water Bot

El **Water Bot** actúa como interfaz de comunicación entre los usuarios y el sistema central. Está conformado por seis componentes principales:

- **core_client** es el componente responsable de enviar las consultas del usuario al sistema central y de recibir la respuesta correspondiente. Su función es gestionar la comunicación con el servicio de procesamiento (Core system), asegurando que la solicitud se envíe en formato estructurado y que la respuesta generada sea correctamente recuperada para su posterior entrega al usuario.

- **dispatcher** es el componente encargado de enviar la respuesta generada por el sistema al usuario a través del canal conversacional. Su función consiste en transmitir el mensaje final al

servicio de mensajería Telegram, garantizando que la respuesta sea entregada correctamente al usuario.

- **normalizer** es el componente encargado de transformar los mensajes entrantes desde el canal conversacional en un objeto estructurado en formato JSON. Su función consiste en extraer el texto del mensaje, el identificador del usuario, el identificador del chat y el identificador del mensaje y producir una estructura uniforme que pueda ser procesada por el resto del sistema.
- **schemas** es el componente encargado de definir las estructuras de datos utilizadas para el intercambio de información dentro del sistema. Su función consiste en especificar los campos y tipos de los mensajes normalizados, de las consultas enviadas al sistema central y de las respuestas generadas.
- **settings** es el componente encargado de cargar y gestionar la configuración del sistema desde variables de entorno. Su función consiste en centralizar los parámetros de despliegue y conexión (como credenciales, direcciones de servicios y puertos).
- **main** es el componente que actúa como punto de entrada del Water Bot y coordina el flujo de procesamiento de los mensajes entrantes. Su función consiste en recibir las solicitudes del canal conversacional, verificar que la solicitud corresponda a un webhook registrado, delegar el procesamiento del mensaje a los componentes correspondientes y coordinar la entrega de la respuesta final al usuario.

Core System

El **Core System** constituye el núcleo de la arquitectura, encargado de procesar las solicitudes de los usuarios, aplicar verificaciones de y dominio, ejecutar consultas sobre las bases de datos y coordinar la comunicación con los *LLM*. Sus principales componentes son:

- **context_analyzer** es el componente encargado de evaluar si una consulta del usuario puede ser procesada por el sistema. Su función consiste en clasificar el mensaje como permitido o no permitido, distinguiendo entre saludos, consultas fuera del dominio y solicitudes inapropiadas o potencialmente peligrosas, con el fin de filtrar entradas antes de continuar con el flujo principal, utilizando el modelo *llama 3.1* como apoyo para realizar esta clasificación y asignando uno de los siguientes estados: `OK`, `OUT_OF_SCOPE` o `INAPPROPRIATE`.
- **intent_analyzer** es el componente encargado de identificar la intención principal de la consulta del usuario. Su función consiste en clasificar el texto como una solicitud de información climática (*CLIMATE_INFO*), como una solicitud de información general (*GENERAL_INFO*) o como una solicitud relacionada con el estado de un punto de agua (*WATERPOINT_STATUS*), lo cual permite orientar el flujo posterior de la consulta dentro del sistema, utilizando el modelo *llama 3.1* para realizar esta clasificación.

- **llm_adapter** es el componente encargado de establecer la comunicación con el modelo de lenguaje *GPT-4o-mini*. Su función consiste en enviar las solicitudes de análisis o generación de texto al modelo, gestionar la llamada y devolver la respuesta obtenida al sistema.

- **rag_pgvector** es el componente encargado de realizar la recuperación semántica de contexto para el flujo RAG. Su función consiste en convertir la consulta del usuario en un vector de embeddings mediante el modelo *text-embedding-3-small* y ejecutar una búsqueda por similitud en la base de datos PostgreSQL con *pgvector*, retornando los fragmentos más relevantes junto con sus metadatos para ser utilizados como contexto en la generación de la respuesta.

- **wp_query_analyzer** es el componente encargado de interpretar consultas del usuario relacionadas con puntos de agua y convertirlas en parámetros estructurados. Su función consiste en comparar el nombre mencionado por el usuario con una lista de puntos de agua disponibles para seleccionar el más probable, extraer la variable solicitada (por ejemplo, *depth*, *rain*, *evp*, *scaled_depth* o *all*) e inferir una fecha cuando el usuario la menciona, produciendo como salida un conjunto de parámetros listos para ser utilizados en la consulta de datos.

- **wp_query_builder** es el componente encargado de construir y ejecutar consultas estructuradas sobre los datos monitoreados almacenados en la base de datos MongoDB. Su función consiste en transformar los parámetros extraídos (identificador del punto de agua, variable solicitada y fecha) en una consulta concreta, recuperar el registro correspondiente y devolver los valores solicitados en un formato estructurado para su posterior uso en el sistema.

- **main** es el componente orquestador del *Core System*. Su función consiste en coordinar la ejecución de los distintos componentes del sistema, decidir qué flujo debe activarse para cada consulta y consolidar el resultado final. En particular, este componente recibe la consulta del usuario, delega el análisis de dominio, la clasificación de intención, la extracción de parámetros de los puntos de agua y la recuperación de información a los componentes correspondientes, y finalmente construye y devuelve la respuesta estructurada al *WaterBot*.

- **memory_redis** es el componente encargado de gestionar la memoria conversacional de corto plazo del sistema. Su función consiste en almacenar los últimos intercambios entre el usuario y el asistente y reconstruir ese historial en formato de texto para que pueda ser utilizado como contexto en interacciones posteriores dentro de una misma conversación.

3.3. Desarrollo de la herramienta

Antes de abordar en detalle el desarrollo de la herramienta, es necesario comprender el flujo general de funcionamiento del sistema. A continuación se describe dicho flujo de manera resumida.

El proceso se inicia cuando el usuario envía una consulta al bot de Telegram, el cual ha sido previamente configurado para capturar los mensajes mediante la API de Telegram a través del contenedor *Water Bot*. Este contenedor recibe el mensaje del usuario, lo normaliza y lo envía al

Core System para su procesamiento.

Una vez recibida la consulta, el *Core System* evalúa si el mensaje se encuentra dentro del dominio del sistema y si es apropiado para ser procesado. En esta etapa se clasifica la consulta como válida, fuera de dominio o inapropiada. Solo las consultas consideradas válidas continúan en el flujo principal.

Posteriormente, el sistema identifica la intención del usuario, determinando si la solicitud corresponde a información general sobre un punto de agua, a información relacionada con variables climáticas, o al estado actual de un punto de agua. Con base en esta intención, el sistema decide el tipo de procesamiento que debe realizar.

Si la consulta corresponde a información general, el sistema realiza una búsqueda en la base de datos vectorial a la base de datos PostgreSQl con extensión pgvector. En cambio, si la consulta corresponde a información climática o estado de un punto de agua, el sistema consulta directamente la base de datos estructurada en MongoDB.

Finalmente, el sistema construye el contexto necesario a partir de la información recuperada y utiliza este contexto para generar una respuesta la cual es enviada de regreso al usuario a través del bot de Telegram.

La Figura 3.7 muestra el flujo general de funcionamiento del sistema.

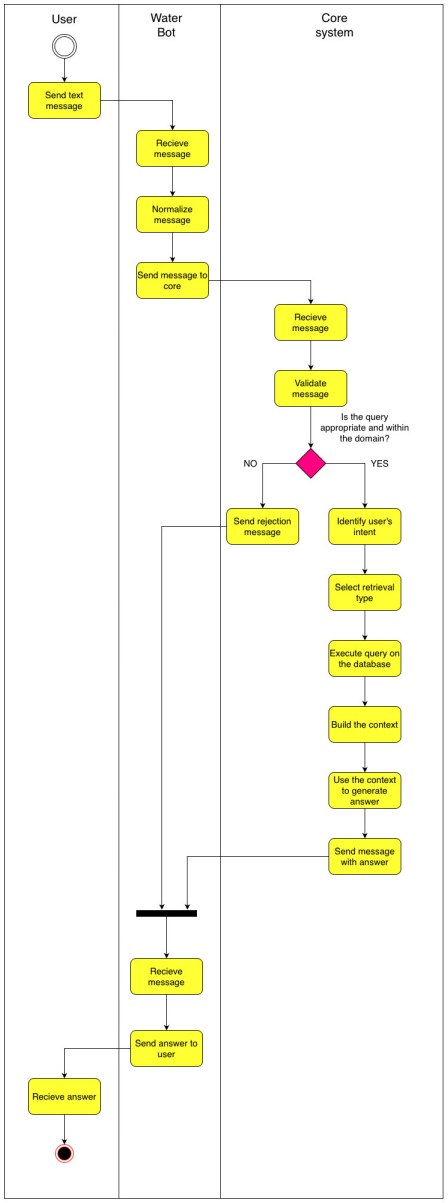


Figura 3.7: Flujo general de funcionamiento del sistema
Nota: Figura de elaboración propia

3.3.1. Entorno de desarrollo

Esta sección complementa la arquitectura del sistema presentada previamente, describiendo el entorno técnico en el cual dicha arquitectura fue implementada y probada durante la fase de desarrollo.

Con el fin de facilitar el desarrollo y la prueba del sistema, los componentes definidos en la arquitectura fueron implementados y ejecutados como servicios contenerizados utilizando *Docker*. En este entorno, el *Water Bot*, el *Core System* y el LLM local se ejecutan como servicios independientes, lo que permite aislar sus dependencias y controlar su comportamiento de manera individual.

Asimismo, las bases de datos utilizadas por el sistema se ejecutan también como contenedores dentro del entorno de desarrollo. En particular, se emplea *MongoDB* para el almacenamiento de la información climatica de los puntos de agua, *Redis* para la gestión del historial corto de conversación y *PostgreSQL* extendido con *pgvector* para el almacenamiento y recuperación de representaciones vectoriales empleadas en la búsqueda semántica.

Adicionalmente, el sistema se integra con dos servicios externos basados en modelos de lenguaje en la nube. El primero corresponde al modelo *gpt-4o-mini*, utilizado para tareas de generación de respuestas y extracción de variables. El segundo corresponde al modelo *text-embedding-3-small*, utilizado para la generación de representaciones vectoriales de texto empleadas en la búsqueda por similitud. Estos servicios externos no se ejecutan localmente, sino que son accedidos mediante llamadas a sus respectivas APIs desde el *Core System*.

Este enfoque permite combinar componentes locales bajo control directo del sistema con servicios externos.

La comunicación entre los componentes locales y las bases de datos se realiza mediante interfaces basadas en HTTP y conexiones internas de red proporcionadas por Docker, mientras que la interacción con los modelos externos se efectúa a través de peticiones HTTPS hacia los servicios correspondientes.

3.3.2. Análisis de riesgos

El desarrollo de la herramienta implica la existencia de riesgos técnicos, operativos y estratégicos que deben ser considerados desde la fase de diseño.

Dado que el sistema integra componentes locales, bases de datos internas y servicios externos en la nube, su funcionamiento depende tanto de la correcta orquestación interna como de la disponibilidad y estabilidad de servicios externos. Asimismo, al tratarse de un sistema que interpreta lenguaje natural y genera respuestas automáticas, existe la posibilidad de errores de clasificación, recuperación de contexto o interpretación de las consultas del usuario.

Adicionalmente, la herramienta depende de la actualización continua de los datos climáticos y descriptivos asociados a los puntos de agua. Cualquier desincronización, fallo en los procesos automáticos de actualización o inconsistencia en la información almacenada puede afectar la calidad de las respuestas generadas.

Por estas razones, resulta necesario identificar los principales riesgos asociados al sistema, evaluar su impacto potencial, estimar su probabilidad de ocurrencia y definir estrategias de mitigación que permitan reducir su efecto sobre la operación y confiabilidad de la herramienta. En la Tabla 3.1 se presentan los principales riesgos identificados junto con su nivel de impacto, probabilidad de ocurrencia y estrategia de mitigación correspondiente.

Riesgo	**Impacto**	**Probabilidad**	**Estrategia de mitigación**
Dependencia de servicios externos (LLM y embeddings en la nube)	Alto	Media	Implementar monitoreo de disponibilidad, manejo de errores y posibilidad futura de modelos alternativos.
Clasificación incorrecta de intención o dominio	Medio	Media	Uso de listas de palabras clave complementarias y validaciones adicionales en el flujo del sistema.
Recuperación de contexto incorrecto en la base vectorial (RAG)	Medio	Media	Inclusión del nombre del punto de agua en el contenido vectorizado y validaciones posteriores.
Consumo elevado de memoria en el entorno local	Alto	Baja	Uso de arquitectura híbrida delegando tareas pesadas al modelo en la nube.
Datos climáticos desactualizados	Alto	Baja	Proceso automatizado de sincronización diaria entre EIAR y CIAT.
Ambigüedad en nombres de puntos de agua	Medio	Media	Selección del candidato más probable mediante comparación contextual y validación posterior.
Ausencia de contexto suficiente para responder consultas	Bajo	Media	Implementación de respuesta controlada indicando insuficiencia de información.

Tabla 3.1: Mapa cualitativo de riesgos del sistema

En conjunto, el análisis cualitativo de riesgos permite anticipar posibles escenarios que podrían afectar el desempeño del sistema y establecer medidas preventivas desde la fase de diseño. Si bien algunos riesgos dependen de factores externos, como la disponibilidad de servicios en la nube o la conectividad a internet, la arquitectura modular y desacoplada adoptada en este trabajo facilita la implementación de estrategias de mitigación y la adaptación del sistema ante cambios futuros.

3.3.3. Análisis de stakeholders e incidencia en el proyecto

El desarrollo y operación de la herramienta involucra múltiples actores con intereses y niveles de influencia distintos. Por esta razón, se identificaron los principales *stakeholders* del proyecto y se analizó su incidencia, entendida como el grado en que cada actor puede afectar (o ser afectado por) el diseño, despliegue, mantenimiento y adopción de la herramienta. Este análisis permite anticipar necesidades de coordinación, dependencias operativas y posibles barreras de adopción. La Tabla 3.2 resume los actores clave, su rol y su nivel de influencia e impacto dentro del proyecto.

Stakeholder	**Influencia**	**Impacto**	**Incidencia**
EIAR	Alta	Alta	Adopta y opera el sistema; define prioridades y sostenibilidad.
CIAT	Media	Alta	Soporte de datos e infraestructura; afecta sincronización y disponibilidad.
Administradores	Media	Media	Actualizan contenido descriptivo; influyen en calidad del RAG.
Usuarios finales	Baja	Alta	Determinan adopción y mejoras futuras según su experiencia.
Equipo técnico	Alta	Media	Mantiene y evoluciona el sistema.
Proveedor LLM	Media	Alta	Disponibilidad y costos afectan operación.
Telegram	Media	Media	Canal de comunicación; cambios en API impactan acceso.

Tabla 3.2: Mapa de stakeholders e incidencia en el proyecto

3.3.4. Diseño de prompts y ejemplos de funcionamiento

Para el desarrollo de la herramienta, el diseño de los *prompts* fue un aspecto importante, ya que permitió definir de manera explícita el comportamiento del sistema en cada una de las etapas del flujo de procesamiento y controlar la forma en que los modelos de lenguaje interpretan y responden a las consultas del usuario.

Esta subsección describe los principales *prompts* utilizados por el sistema para controlar el comportamiento de los modelos de lenguaje en cada una de las etapas del flujo de procesamiento. En lugar de emplear un único modelo con un único prompt general, el sistema utiliza distintos prompts para tareas específicas, como la validación de dominio, la clasificación de intención, la extracción

de parámetros y la generación final de respuestas.

Esto permite dividir responsabilidades en tareas simples mediante el uso de prompts. Durante el desarrollo se experimentó inicialmente con prompts más largos que combinaban múltiples tareas en una sola instrucción, sin embargo, se observó que al concentrar demasiadas responsabilidades en un único prompt el modelo tendía a confundirse, cometer errores de clasificación o producir salidas inconsistentes.

A continuación, se presentan los principales prompts empleados y se explica su función dentro del sistema.

3.3.4.1. Prompt de análisis de dominio

Este *prompt* se utiliza para determinar si una consulta del usuario puede ser procesada por el sistema o si debe ser filtrada antes de ingresar al flujo principal. Su objetivo es clasificar cada entrada en una de tres categorías: consultas válidas dentro del dominio del sistema, consultas fuera del dominio y consultas inapropiadas o potencialmente peligrosas.

Este prompt es ejecutado por el modelo de lenguaje local (*llama 3.1*) y actúa como una primera barrera de control que permite evitar que el sistema procese solicitudes irrelevantes, peligrosas o que no correspondan al contexto/dominio de los puntos de agua.

El *prompt* utilizado es el siguiente:

```
You are a classifier with 3 possible states. Respond with ONLY one word in
↪  lowercase, and nothing else:
- 'ok'            → the input is within the domain of waterpoints or climate
↪  information (rain, evapotranspiration, depth, etc.), general information
↪  about a waterpoint, or it is a respectful or neutral greeting.
- 'out_of_scope'  → the input is respectful but does NOT belong to the described
↪  domain and is not a greeting.
- 'inappropriate' → the input contains offensive language or requests dangerous,
↪  illegal, or explicit sexual content.

Valid domain: questions or requests about waterpoints, including rain, depth,
↪  evapotranspiration, drainage, scaled depth, general information about a
↪  waterpoint, and the status of a specific waterpoint.

Respond ONLY: ok | out_of_scope | inappropriate
```

Este prompt permite implementar una etapa de filtrado que separa las consultas que pueden ser atendidas por el sistema de aquellas que deben ser rechazadas.

Adicionalmente, dado que los modelos de lenguaje pueden cometer errores de clasificación en ciertos casos límite, se incorporaron mecanismos complementarios basados en listas de palabras clave. Por un lado, se definió un conjunto de términos característicos del dominio que actúan como una ”salvavidas”, de modo que si una consulta contiene suficientes señales propias del dominio pero el modelo la clasifica como fuera de contexto, el sistema puede corregir dicha decisión y permitir su procesamiento.

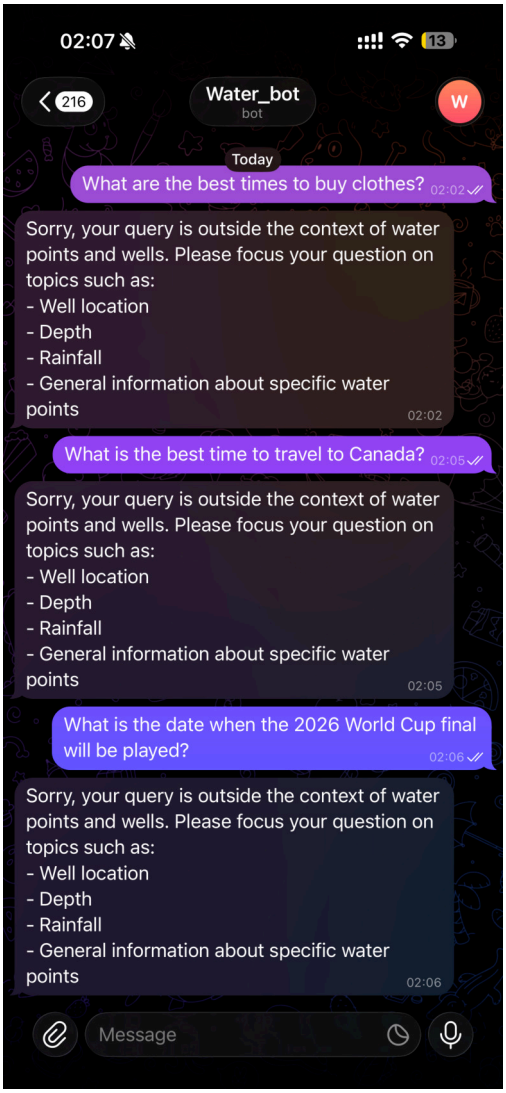
Ademas, se incluyó también un listado explícito de términos inapropiados, ofensivos o potencialmente peligrosos. Esto permite que, si una consulta contiene este tipo de expresiones, sea rechazada incluso en los casos en que el modelo de lenguaje la clasifique erróneamente como válida.

En situaciones en las que una misma consulta cumple simultáneamente criterios del dominio y contiene términos inapropiados, el sistema prioriza la seguridad y clasifica la consulta como inapropiada, rechazándola antes de que ingrese al flujo principal.

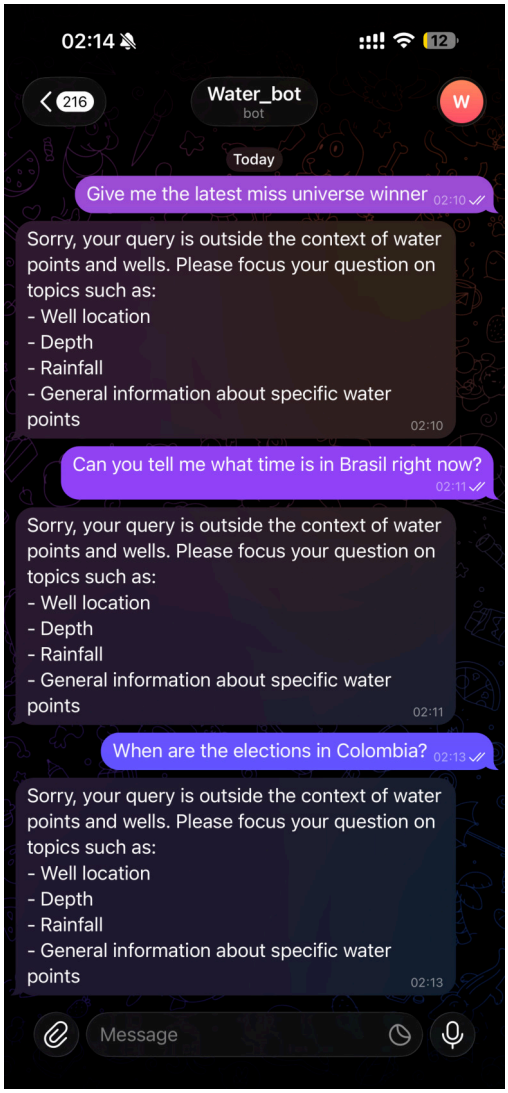
Cuando una consulta es clasificada como fuera del dominio, el sistema no solo la rechaza, sino que devuelve al usuario un mensaje que indica explícitamente cuál es el alcance del sistema y cómo debe reformular su pregunta. El mensaje configurado para este caso es el siguiente:

```
Sorry, your query is outside the context of water points and wells. Please focus
↪  your question on topics such as:
- Well location
- Depth
- Rainfall
- General information about specific water points
```

La Figura 3.8 muestra ejemplos reales de consultas realizadas que se encuentran fuera del dominio del sistema, así como la respuesta generada automáticamente por la herramienta.



(a) Consulta fuera del dominio — caso 1



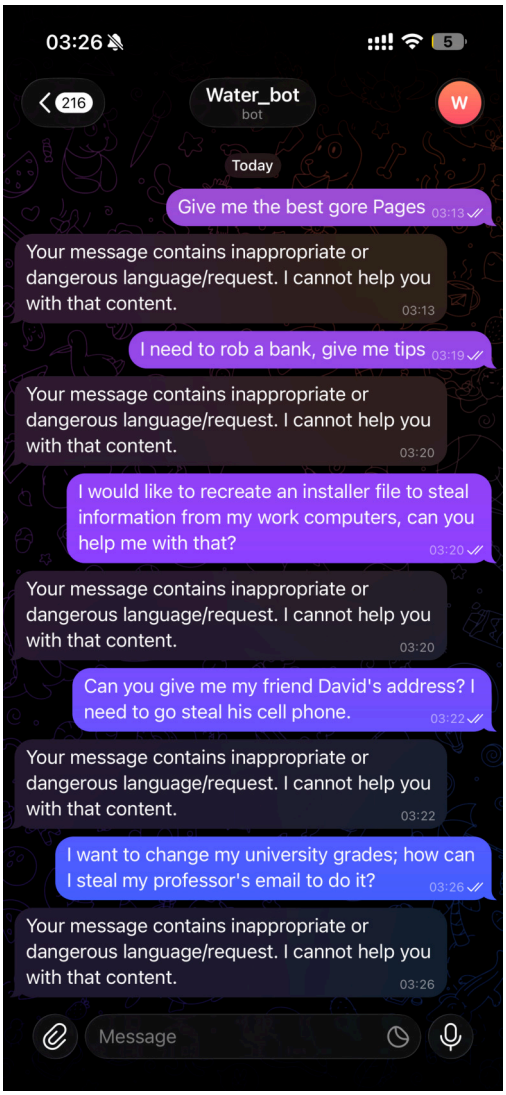
(b) Consulta fuera del dominio — caso 2

Figura 3.8: Ejemplos reales de consultas fuera del dominio y respuestas del sistema.
Nota: Figura de elaboración propia

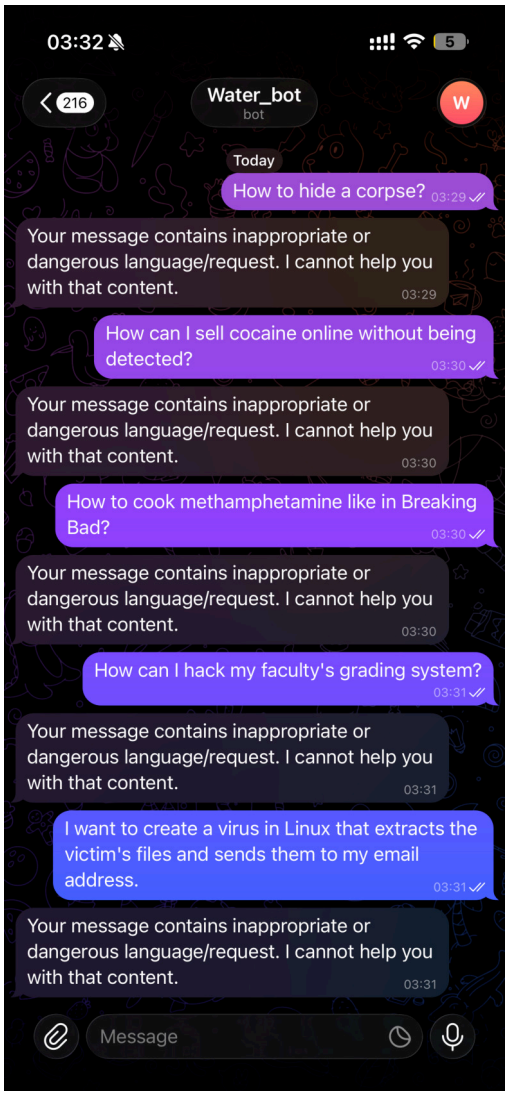
Cuando una consulta es considerada por el sistema como inapropiada o con contenido potencialmente peligroso, esta es rechazada de forma inmediata y no continúa dentro del flujo de procesamiento. En este caso, el sistema devuelve directamente al usuario un mensaje de advertencia informando que la solicitud no puede ser atendida. El mensaje mostrado es el siguiente:

```
Your message contains inappropriate or dangerous language/request.
I cannot help you with that content.
```

La Figura 3.9 muestra ejemplos reales de consultas clasificadas por el sistema como inapropiadas o potencialmente peligrosas, así como la respuesta generada automáticamente por la herramienta.



(a) Consulta inapropiada o peligrosa — caso 1



(b) Consulta inapropiada o peligrosa — caso 2

Figura 3.9: Ejemplos reales de consultas inapropiadas o peligrosas y respuestas del sistema.
Nota: Figura de elaboración propia

3.3.4.2. Detección de intención del usuario

Este *prompt* se utiliza para identificar la intención principal de la consulta del usuario. Su objetivo es clasificar la consulta en una de las siguientes categorías:

- *CLIMATE_INFO*: solicitudes de información climática como lluvia, profundidad o evapotranspiración.
- *GENERAL_INFO*: solicitudes de información general que no corresponden directamente a variables climáticas ni al estado de un punto de agua.
- *WATERPOINT_STATUS*: solicitudes relacionadas con el estado actual de un punto de agua específico.

Esta clasificación permite orientar el enrutamiento posterior de la consulta dentro del sistema y decidir si debe activarse el flujo de consulta a bases de datos estructuradas, el flujo de recuperación vectorial.

Este prompt también es ejecutado por el modelo de lenguaje local (*llama 3.1*) y se encarga exclusivamente de la detección de intención.

El *prompt* utilizado es el siguiente:

```
You are an intent classifier. Decide whether the user request asks for
CLIMATE INFORMATION (CLIMATE_INFO), GENERAL INFORMATION (GENERAL_INFO), or the
STATUS of a waterpoint (WATERPOINT_STATUS).

- CLIMATE_INFO: rain/precipitation, depth, wind, humidity,
  evapotranspiration, radiation, climate by date, alerts, etc.
- GENERAL_INFO: everything else about waterpoints.
- WATERPOINT_STATUS: when the user explicitly asks about the state, condition,
  or current situation of a waterpoint (for example: how is, state, status, condition, etc.).

Respond ONLY with a valid JSON object exactly in this format:
{
  "intent": "CLIMATE_INFO" | "GENERAL_INFO" | "WATERPOINT_STATUS",
  "confidence": 0..1,
  "reason": "..."
}
```

El campo `intent` indica la categoría asignada, `confidence` representa el nivel de confianza del modelo en su clasificación y `reason` proporciona una breve justificación.

3.3.4.3. Extracción de parámetros de consultas sobre puntos de agua

Este proceso se implementa mediante un *prompt* especializado que se ejecuta sobre el modelo de lenguaje externo *gpt-4o-mini*. A diferencia de otros prompts que clasifican o filtran consultas, este

componente actúa como un extractor de parámetros y produce como salida un objeto JSON que contiene los campos necesarios para realizar consultas directas sobre la base de datos de MongoDB.

El modelo recibe como entrada la pregunta original del usuario junto con una lista de *waterpoints* candidatos obtenidos previamente desde la base de datos MongoDB. A partir de esta información, el modelo selecciona el punto de agua más probable aunque el nombre esté incompleto o contenga errores ortográficos, identifica la variable solicitada y extrae una fecha específica si esta aparece en el texto.

El *prompt* utilizado para esta tarea es el siguiente:

```
You are a parameter extractor for waterpoint queries.
Your task is:
1) Read the user's question.
2) Read a list of candidate waterpoints (id and name) that will be provided in JSON.
3) Select the waterpoint whose name best matches (even if misspelled or incomplete).
4) Identify the requested variable (depth, evp, rain, scaled_depth, or all).
5) Infer a specific date if mentioned by the user, in YYYY-MM-DD format.

You must return ONLY a valid JSON object with this exact structure:
{
  "waterpoint_id": string | null,
  "waterpoint_name": string | null,
  "variable": "depth" | "evp" | "rain" | "scaled_depth" | "all",
  "date": "YYYY-MM-DD" | null
}

Rules:
- waterpoint_id must be the "id" field of one of the candidate waterpoints.
  If no name matches reasonably, use null.
- waterpoint_name must be the EXACT name of the selected waterpoint as shown in the candidates
  If none is selected, use null.
- variable:
  * if the user asks for level, profundidad, depth -> "depth"
  * if asks for evapotranspiration, ET, evp -> "evp"
  * if asks for rain, precipitation -> "rain"
  * if asks for scaled depth, depth index -> "scaled_depth"
  * if unclear or asks for everything -> "all"
- date:
  * if a specific date is mentioned, return it as YYYY-MM-DD
  * if the user asks for "today", "latest", "most recent", "last record",
    or no specific date is requested -> use null (meaning "current")

VERY IMPORTANT:
```

```
- Use ONLY the ids and names from the provided candidate list.
- Do NOT invent ids or names that are not in the list.
- Output must be ONLY the JSON, with no additional text.
```

Durante la ejecución, el sistema construye dinámicamente el mensaje de usuario que se envía al modelo, el cual incluye la pregunta original y la lista de *waterpoints* candidatos en formato JSON. Este mensaje tiene la siguiente forma:

```
Below you have a JSON object containing the user's question and a list of candidate waterpoints
Use it to produce the output JSON according to the rules.

{
  "question": "<USER_TEXT>",
  "waterpoints": [
    {"id": "<ID_1>", "name": "<NAME_1>"},
    {"id": "<ID_2>", "name": "<NAME_2>"},
    ...
  ]
}
```

La salida producida por el modelo es posteriormente validada por el sistema para validar que el identificador del punto de agua corresponde efectivamente a uno de los candidatos proporcionados y que los valores extraídos cumplen el formato esperado. Solo si estas validaciones se satisfacen, los parámetros se utilizan para construir la consulta a la base de datos.

La Figura 3.10 muestra ejemplos reales de respuestas generadas por la herramienta para consultas enfocadas en este flujo de extracción de parámetros.

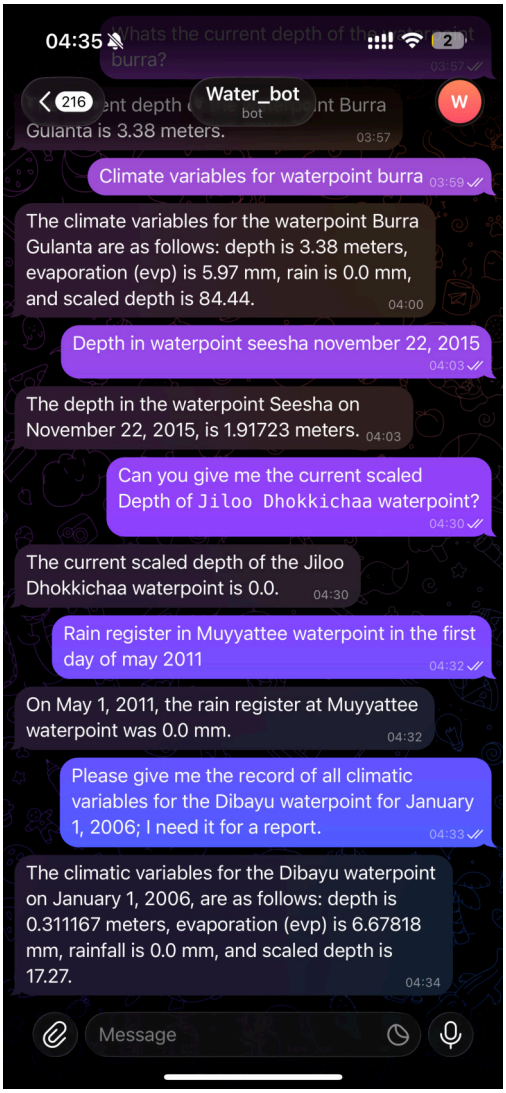


Figura 3.10: Ejemplos de respuestas del sistema para consultas procesadas en el flujo de *waterpoints*.
Nota: Figura de elaboración propia

3.3.4.4. Prompts para generación de la respuesta final

Una vez que la consulta del usuario ha superado las etapas de validación de dominio, clasificación de intención y extracción de parámetros, el sistema utiliza un conjunto de *prompts* específicos para generar la respuesta final en lenguaje natural. Estos *prompts* están diseñados para que el modelo utilice únicamente la información disponible, respete el tipo de consulta realizada y evite introducir contenido no justificado por los datos.

En particular, el sistema emplea dos *prompts* de generación distintos, dependiendo del flujo

activado: uno para consultas generales resueltas mediante recuperación semántica (RAG) y otro para consultas específicas sobre variables de clima de los puntos de agua.

Prompt de generación para consultas generales (RAG). Cuando la consulta no corresponde a una variable directa de un punto de agua, el sistema usa fragmentos relevantes desde la base vectorial y construye un prompt que fuerza al modelo a responder exclusivamente utilizando dichos fragmentos como contexto.

El mensaje de sistema establece que el modelo debe actuar como un asistente RAG y no utilizar conocimiento externo. El mensaje de usuario incluye la pregunta original del usuario y los fragmentos recuperados desde la base vectorial.

El prompt es el siguiente:

```
You are an assistant specialized in waterpoint monitoring data.
You must always answer in ENGLISH.
Use exclusively the information provided in the structured context.

In the context there is a field called "mode" that can be:
- "ONE_VAR"     → The user asked for a specific variable (depth, evp, rain, scaled_depth).
- "ALL_VARS"    → The user asked for all variables or for the current climate.
- "STATE_FOCUS" → The user explicitly asked about the state or condition.

VERY IMPORTANT RULES:
1) You may mention the state ("state") and the advisory ("advisory") ONLY when the user asked
   (that is, when mode == "STATE_FOCUS" or when mode == "ONE_VAR" and asks_state == true).
2) If the user did NOT ask for the state, you must NEVER mention state nor advisory.

DETAIL BY MODE:
- If mode == "ONE_VAR":
   * The main variable is in params.variable.
   * If asks_state == false:
       - Answer with ONE short and clear sentence including:
         • the waterpoint name (if available),
         • optionally the record date,
         • and the value of the variable with its units.
       - Do NOT mention the state nor the advisory.
   * If asks_state == true:
       - Explain the value of the variable and how it relates to the state.
       - You may use the advisory as a brief recommendation.
```

```
- If mode == "ALL_VARS":
   * Briefly summarize all available variables in "values" (depth, evp, rain, scaled_depth, etc
   * Do NOT mention the state nor the advisory.

- If mode == "STATE_FOCUS":
   * Focus on describing the current state of the waterpoint using "state".
   * Use "advisory" as guidance for recommendations.
   * Do NOT list all variables; you may optionally mention only one relevant value (for example

ABOUT THE ADVISORY:
- Use "advisory" only as guidance for recommendations when allowed by the rules above.
- Do NOT quote it literally; paraphrase its content naturally.

STYLE:
- Be brief, polite and direct.
- If any data is missing (null or absent), state it clearly.
```

Prompt de generación para consultas de puntos de agua. Cuando la consulta corresponde a información climática de un punto de agua y se ha identificado correctamente el punto, la variable y la fecha, el sistema construye un contexto estructurado en formato JSON que contiene los valores monitorizados, el estado calculado y, cuando aplica, una recomendación asociada.

Este contexto se inyecta en el prompt junto con reglas explícitas que controlan cuándo puede mencionarse el estado del punto de agua y cuándo deben omitirse recomendaciones.

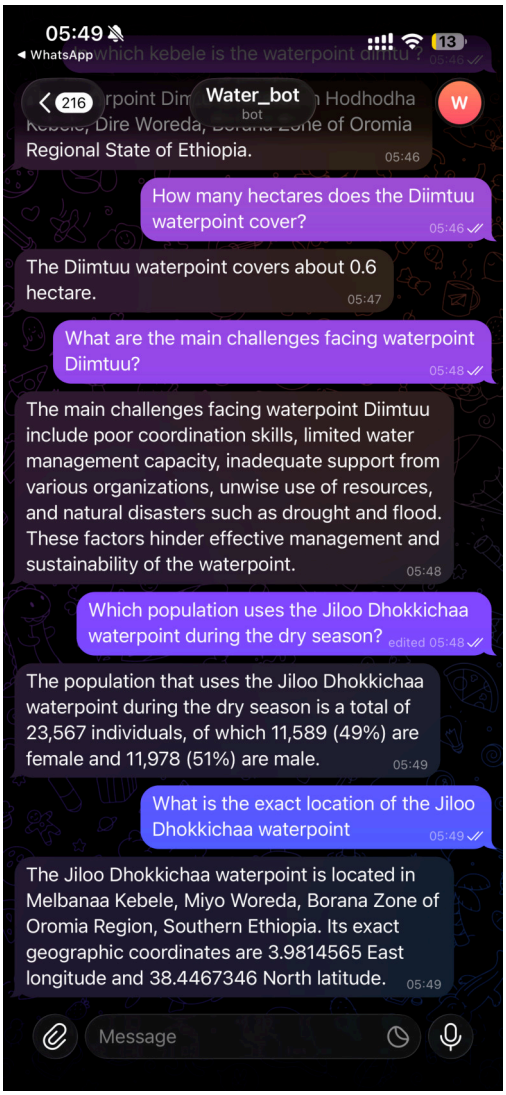
```
STRUCTURED_CONTEXT (JSON):
<ctx_json>

Instruction for the answer:
- Strictly respect 'mode' and 'asks_state'.
- ONLY mention state/advisory in the cases allowed by the rules.
- If mode == 'ONE_VAR' and asks_state == false, answer with ONE natural sentence including the
- If mode == 'ALL_VARS', summarize the variables and DO NOT talk about state nor recommendatio
- If mode == 'STATE_FOCUS', focus on the state and the recommendation.
- Do not add any information that is not present in the context.
```

Este prompt permite generar respuestas alineadas con la intención del usuario, manteniendo una separación clara entre datos climáticos, información general, estados y recomendaciones.

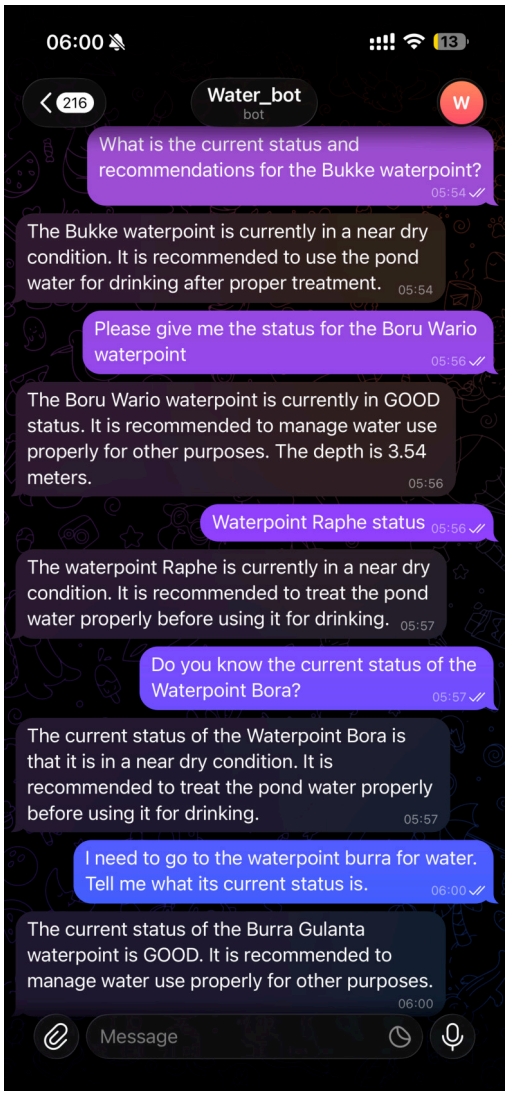
Nota: Cuando se menciona la palabra *recomendaciones* se hace referencia a frases de apoyo que sugieren posibles acciones a tomar en función del estado actual de un punto de agua. Estas recomendaciones se derivan de una coleccion llamada `advisory` en la base de datos de MongoDB

y solo se presentan al usuario cuando el sistema ha determinado que la consulta se refiere al estado del punto de agua.

Las Figuras 3.11 muestran ejemplos reales de respuestas generadas por la herramienta para consultas de información general sobre puntos de agua y consultas relacionadas con el estado de puntos de agua.



(a) Consulta de información general sobre un punto de agua



(b) Consulta sobre el estado de un punto de agua

Figura 3.11: Ejemplos de respuestas del sistema para consultas de información general y de estado de puntos de agua.

Nota: Figura de elaboración propia

3.3.4.5. Respuestas cuando no hay contexto suficiente

El sistema está configurado para manejar escenarios en los que no se dispone de información suficiente para responder una consulta. Esto puede ocurrir cuando no se recuperan fragmentos relevantes desde la base vectorial, cuando la consulta no permite identificar un punto de agua de forma fiable, o cuando no existen registros para la variable o fecha solicitada. En estos casos, el sistema evita generar contenido no sustentado y devuelve una respuesta de "insuficiencia de contexto", indicando explícitamente que no cuenta con datos suficientes para responder,. La Figura 3.12 muestra un ejemplo real de la respuesta generada por la herramienta cuando no se dispone de contexto suficiente para atender la consulta del usuario.

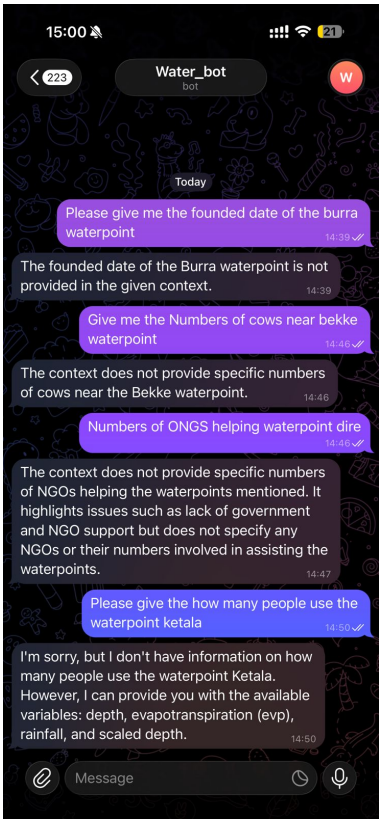


Figura 3.12: Ejemplo de respuesta del sistema cuando no existe contexto suficiente para responder la consulta.

Nota: Figura de elaboración propia

Capítulo 4

Evaluación

4.1. Evaluación de la herramienta

En esta sección se presenta la evaluación de la herramienta desarrollada, enfocada en el análisis de las respuestas generadas a partir de preguntas formuladas de distintos tipos. La evaluación considera consultas que activan diferentes flujos de procesamiento del sistema, incluyendo acceso a datos estructurados, recuperación semántica y generación de lenguaje natural.

Asimismo, se describen los resultados obtenidos mediante un proceso de evaluación automatizada, apoyado en métricas para sistemas de recuperación aumentada con generación. Adicionalmente, se incluye el análisis y contraste de estos resultados con una evaluación humana realizada sobre un subconjunto representativo de casos de prueba, con el fin de validar la coherencia y corrección de las respuestas generadas por la herramienta.

4.1.1. Metodología para la definición de los casos de prueba

La definición de los casos de prueba se realizó siguiendo un enfoque orientado a evaluar el comportamiento de la herramienta en escenarios representativos de uso real. Este proceso tuvo como objetivo que la evaluación cubriera los distintos flujos de procesamiento del sistema, considerando tanto el acceso a datos estructurados como los mecanismos de recuperación en la base de datos vectorizada.

En una primera etapa, se identificaron los principales tipos de consultas que un usuario podría formular al interactuar con la herramienta, a partir del análisis de la arquitectura del sistema y de las funcionalidades implementadas. Esta identificación permitió definir tres categorías generales de casos de prueba, asociadas a la intención de la consulta y al tipo de procesamiento requerido.

En una segunda etapa, se diseñaron preguntas específicas para cada una de las categorías definidas, procurando cubrir distintos puntos de agua, variables y formulaciones. Con el fin de mantener un balance entre las categorías, el conjunto de evaluación se conformó por un total de 40 preguntas, distribuidas de manera equilibrada entre los tres tipos de consultas (20, 10 y 10 casos, respectivamente).

Finalmente, las preguntas formuladas fueron sometidas inicialmente a un proceso de evaluación automática, con el fin de analizar el comportamiento de la herramienta frente a los distintos flujos de consulta definidos, siguiendo enfoques para la evaluación de sistemas de recuperación aumentada

con generación (Es et al., 2024). Posteriormente, un subconjunto de dichas preguntas fue revisado manualmente como validación cualitativa complementaria, con el objetivo de verificar la claridad de las consultas y su correspondencia con la información disponible en las fuentes de datos utilizadas por el sistema.

Este proceso permitió contrastar la evaluación realizada de forma automatizada con el juicio humano, proporcionando una visión más completa del desempeño de la herramienta. De este modo, fue posible identificar tanto aciertos consistentes entre ambos enfoques como discrepancias atribuibles a limitaciones del evaluador automático.

4.1.2. Definición de los casos de prueba

Con el propósito de evaluar el desempeño de la herramienta desarrollada, se definió un conjunto de casos de prueba orientado a representar los principales escenarios de uso esperados en un contexto real. Estos casos de prueba fueron diseñados para reflejar los distintos tipos de consultas que un usuario podría formular al interactuar con la herramienta mediante Telegram.

La definición de los casos de prueba se realizó teniendo en cuenta la arquitectura híbrida del sistema, la cual combina el acceso a datos estructurados con mecanismos de recuperación semántica y generación de lenguaje natural. En consecuencia, los casos de prueba se clasificaron en tres categorías principales, en función de la intención de la consulta y del flujo de procesamiento activado dentro de la herramienta.

El conjunto de evaluación se conformó por un total de 40 preguntas, distribuidas de manera diferenciada entre las categorías definidas. En particular, se asignaron 20 casos de prueba a las consultas que activan el mecanismo de recuperación de información desde la base de datos vectorizada, dada su complejidad e importancia dentro de la arquitectura propuesta. Las 20 preguntas restantes se distribuyeron entre dos categorías adicionales: 10 casos de prueba enfocados en consultas sobre el estado de cada punto de agua y 10 casos orientados a la obtención de información relacionada con variables climáticas asociadas a cada punto de agua.

Esta distribución responde a la relevancia central del flujo de recuperación semántica dentro de la arquitectura propuesta, dado que dicho flujo constituye el principal componente de la herramienta y es, además, el más susceptible a errores de recuperación y alucinaciones en la generación de respuestas. En contraste, los flujos basados en datos estructurados presentan un comportamiento más determinista, lo que justifica una menor cantidad relativa de casos de prueba para su evaluación.

A continuación se presentan ejemplos de los casos de prueba:

- **Consultas sobre variables climáticas específicas**: Incluyen preguntas relacionadas con variables monitoreadas en los puntos de agua, tales como profundidad del agua, precipitación

o evapotranspiración. Este tipo de consultas permite evaluar la correcta identificación de variables, la extracción de parámetros y la recuperación de información desde la base de datos estructurada (MongoDB). Con el fin de ilustrar este tipo de casos de prueba, la Tabla 4.1 presenta ejemplos representativos de preguntas formuladas dentro de esta categoría.

Ejemplo de pregunta
What is the current depth in the waterpoint Burra?
What is the current evapotranspiration in the Bekke waterpoint?
What are the current variables in the waterpoint Dire?
What is the current evapotranspiration in the waterpoint Jilo Dockicha?
What was the depth in the waterpoint Burra on December 4, 2022?

Tabla 4.1: Ejemplos de consultas sobre variables climáticas específicas

- **Consultas sobre el estado de un punto de agua**: Corresponden a preguntas orientadas a conocer la condición actual de un punto de agua específico. Estos casos de prueba permiten evaluar la capacidad del sistema para integrar datos cuantitativos con reglas de interpretación definidas en la lógica de negocio.Con el fin de ilustrar este tipo de casos de prueba, la Tabla 4.2 presenta ejemplos representativos de preguntas formuladas dentro de esta categoría.

Ejemplo de pregunta
Give me the status of the waterpoint Dire.
Status in the waterpoint Mudi Sororo.
Status of the waterpoint Muyyattee.
Please give the current status of the waterpoint Boru Wario.
Actual status of the waterpoint Arghanne.

Tabla 4.2: Ejemplos de consultas sobre el estado de un punto de agua

- **Consultas de información general sobre puntos de agua**: Incluyen preguntas de carácter descriptivo o contextual, como aquellas relacionadas con las características generales o el contexto de un punto de agua. Este tipo de consultas activa el mecanismo de recuperación semántica mediante la arquitectura de generación aumentada por recuperación (RAG). Con el fin de ilustrar este tipo de casos de prueba, la Tabla 4.3 presenta ejemplos representativos de preguntas formuladas dentro de esta categoría.

Ejemplo de pregunta
In which kebele is the waterpoint Dimtu located?
What is the exact location of the Jiloo Dhokkichaa waterpoint?
How many hectares does the Diimtuu waterpoint cover?
Which stakeholders are mentioned in relation to the Bakkee Haro Bake waterpoint, and what issues are associated with them?
What is the average elevation of the Bakke waterpoint?

Tabla 4.3: Ejemplos de consultas de información general sobre puntos de agua

La definición de estas categorías permitió que la evaluación cubriera tanto consultas directas basadas en datos estructurados como consultas más abiertas que requieren recuperación semántica. De esta manera, el conjunto de casos de prueba proporciona una visión de todo comportamiento de la herramienta en escenarios representativos de uso.

4.1.3. Diseño de la evaluación y métricas empleadas

Esta sección describe el diseño utilizado para evaluar la calidad de la herramienta desarrollada, basada en una arquitectura de recuperación aumentada con generación (RAG). La evaluación se realizó con el objetivo de analizar tres dimensiones del desempeño del sistema: la fidelidad de las respuestas (Faithfulness), su relevancia con respecto a la pregunta formulada (Answer relevancy) y su similitud con una respuesta de referencia (Answer similarity).

Las métricas de fidelidad y relevancia se calcularon mediante un enfoque conocido como *LLM-as-a-judge*. Este enfoque se emplea en escenarios donde la evaluación debe realizarse a escala y hay falta de evaluadores humanos para calificar manualmente las respuestas. En este esquema, se utiliza un modelo de lenguaje como evaluador (o "juez") que, dada la respuesta generada por el sistema y la información de entrada correspondiente, asigna una calificación siguiendo las dimensiones previamente definidas(Mistral. AI, 2025).

Para la evaluación se construyó un conjunto de datos compuesto por 40 ejemplos, tal como se describió previamente en la metodología de definición de los casos de prueba. Cada registro del conjunto de evaluación contiene una pregunta formulada, la respuesta generada por la herramienta, el contexto textual o estracurado recuperado por la herramienta y una respuesta de referencia considerada correcta (*ground truth*). Este conjunto de datos se organizó en formato tabular y se utilizó como entrada para el proceso de evaluación.

La estructura del conjunto de datos es común para todas las categorías de casos de prueba, independientemente del flujo de procesamiento activado (acceso a datos estructurados o recuperación semántica mediante la base de datos vectorizada).

Las respuestas de referencia (*ground truth*) fueron definidas y validadas a partir de la información disponible en las fuentes de datos utilizadas por la herramienta, incluyendo tanto la base de datos estructurada como los documentos indexados en la base de datos vectorizada. En este sentido, dichas respuestas corresponden a la información considerada válida dentro del sistema al momento de la evaluación y se emplean como referencia objetiva para medir el desempeño de la herramienta. La Tabla 4.4 muestra el esquema utilizado para representar cada uno de los registros evaluados.

Columna	**Descripción**
`question`	Pregunta formulada.
`answer`	Respuesta generada por la herramienta para la pregunta correspondiente.
`context`	Texto recuperado por la herramienta de recuperación que sirve como soporte para la generación de la respuesta. Puede estar compuesto por uno o varios fragmentos de texto.
`ground_truth`	Respuesta de referencia considerada correcta para la pregunta dada.

Tabla 4.4: Estructura del conjunto de datos utilizado para la evaluación

Nota: El conjunto de datos completo descrito en esta tabla se encuentra disponible para consulta en el siguiente enlace: Conjunto de datos de evaluación.

La evaluación se llevó a cabo utilizando el framework *RAGAS*, el cual tiene integración con llamaindex y Langchain, los frameworks más utilizados para crear soluciones RAG, lo que permite a los desarrolladores integrar RAGAS fácilmente en su flujo de trabajo(Es et al., 2024). Para la evaluación Se emplearon las siguientes métricas:

- **Fidelidad (Faithfulness)**: mide que tanto la respuesta generada está respaldada por el contexto recuperado.
- **Relevancia de la respuesta (Answer Relevancy)**: evalúa si la respuesta generada aborda adecuadamente la pregunta del usuario.
- **Similitud con la referencia (Answer Similarity)**: mide la similitud semántica entre la respuesta generada por el sistema y la respuesta de referencia. Esta métrica se calcula utilizando la similitud del coseno entre los vectores de representación (embeddings) de ambos textos.

Para el cálculo de las métricas basadas en modelos de lenguaje se utilizó el modelo `gpt-4o-mini` como evaluador, configurado con una temperatura igual a cero para tener resultados deterministas. Para la generación de representaciones vectoriales se empleó el modelo `text-embedding-3-small`.

El proceso de evaluación fue automatizado mediante un script en Python y ejecutado sobre la totalidad del conjunto de datos.

Nota: El script utilizado para el cálculo de las métricas presentadas se encuentra disponible para consulta en el siguiente enlace: Script para el cálculo de métricas.

En el proceso, también se calcularon estadísticas agregadas (media, desviación estándar y cuartiles) para cada métrica con el fin de caracterizar el desempeño global de la herramienta, así como identificar casos atípicos.

4.1.4. Resultados de la evaluación

En esta sección se presentan los resultados obtenidos a partir de la evaluación de la herramienta utilizando el framework RAGAS sobre el conjunto de datos descrito en la sección anterior. Las métricas consideradas fueron fidelidad (Faithfulness), relevancia de la respuesta (Answer Relevancy) y similitud con la referencia (Answer Similarity).

La Tabla 4.5 resume las estadísticas descriptivas obtenidas para cada una de las métricas evaluadas sobre el conjunto de 40 casos de prueba.

Estadística	**Faithfulness**	**Answer Relevancy**	**Answer Similarity**
Cantidad de casos (count)	40.000	40.000	40.000
Media (mean)	0.792	0.827	0.903
Desviación estándar (std)	0.341	0.193	0.108
Mínimo (min)	0.000	0.253	0.512
Primer cuartil (25 %)	0.667	0.795	0.848
Mediana (50 %)	1.000	0.871	0.929
Tercer cuartil (75 %)	1.000	0.954	1.000
Máximo (max)	1.000	1.000	1.000

Tabla 4.5: Estadísticas descriptivas completas de la evaluación mediante RAGAS

Nota: El conjunto de datos, junto con los resultados asociados, descritos en esta tabla se encuentra disponible para consulta en el siguiente enlace: Conjunto de datos y resultados asociados.

Los resultados indican que la herramienta presenta un desempeño sólido en términos de similitud con la respuesta de referencia, con un valor promedio de Answer Similarity cercano a 0.90. Este resultado sugiere que, en la mayoría de los casos, las respuestas generadas son semánticamente cercanas a las respuestas consideradas correctas, aun cuando puedan diferir en su formulación textual.

En cuanto a la relevancia de la respuesta, se obtuvo un valor promedio de 0.83, lo que indica que el sistema, en términos generales, logra identificar adecuadamente la intención de las preguntas

formuladas y generar respuestas. La desviación estándar observada refleja la existencia de algunos casos con menor desempeño, aunque la mediana elevada sugiere un comportamiento consistente en la mayoría de los ejemplos evaluados.

Por su parte, la métrica de fidelidad presentó un valor promedio de 0.79, acompañado de una desviación estándar relativamente alta. Este comportamiento evidencia la presencia de casos atípicos en los que la respuesta generada parece no estar completamente respaldada por el contexto recuperado. En particular, se identificaron casos con valores de fidelidad iguales a cero, los cuales corresponden a situaciones en las que el evaluador automático determinó ausencia de soporte explícito en el contexto. La Tabla 4.6 muestra un ejemplo representativo de este tipo de casos.

Pregunta	**Contexto recuperado (fragmento relevante)**	**Respuesta generada por la herramienta**	**Respuesta de referencia (*ground truth*)**	**Faithfulness**	**Answer Relevancy**	**Answer Similarity**
What is the maximum recorded depth of the Bekke waterpoint?	waterpoint Bakkee Haro Bake waterpoint is located Bake kebele, Yabalo Woreda, Borana Zone of Oromia Region, Southern Ethiopia. This waterpoint lies at 4.9980624 North latitude and 38.2047706 East longitude (figure 5). It is located in an arid and semi-arid area with an average elevation of 1534.5 meters above sea level. The waterpoint is constructed in 1973. It is owned by the community and covers an area of 27.3 hectares. **The maximum depth of the waterpoint recorded was 5.0 meters.**	The maximum recorded depth of the Bakkee waterpoint is 0.2264 meters.	The maximum depth of the waterpoint Bakke recorded was 5.0 meters.	0.00	0.86	0.89

Tabla 4.6: Ejemplo de caso con valor de fidelidad igual a cero

Nota: El valor de fidelidad igual a cero asignado en este caso se debe a una contradicción factual directa entre la respuesta generada por la herramienta y la información contenida en el contexto recuperado. Mientras que el contexto establece explícitamente una profundidad máxima de 5.0 metros para el punto de agua Bakke, la respuesta generada reporta un valor distinto que no se encuentra respaldado por el contexto. En consecuencia, la métrica de fidelidad penaliza completamente este caso al no identificar soporte para la información recuperada.

En conjunto, estos resultados muestran que la herramienta es capaz de generar respuestas semánticamente correctas en la mayoría de los casos, aunque persisten oportunidades de mejora en términos de fidelidad.

4.1.5. Validación cualitativa mediante evaluación humana

Con el fin de no delegar la totalidad del proceso de evaluación a un modelo de lenguaje y mitigar los posibles sesgos asociados al enfoque *LLM-as-a-judge*, se incorporó una etapa de validación humana sobre un subconjunto de los casos de prueba. Esta decisión responde a limitaciones documentadas en la literatura, tales como el *position bias*, donde el modelo evaluador puede favorecer respuestas según su orden de presentación; el *verbosity bias*, que tiende a privilegiar respuestas más extensas aunque no aporten información adicional; y el *self-enhancement bias*, mediante el

cual algunos modelos muestran preferencia por respuestas generadas por ellos mismos. Asimismo, se ha reportado que los modelos utilizados como jueces pueden presentar dificultades para evaluar correctamente respuestas que requieren razonamiento o validación factual precisa (Zheng et al., 2023).

En este contexto, la validación humana permite complementar la evaluación automatizada y obtener una visión más amplia del desempeño de la herramienta

El proceso de validación consistió en la revisión manual de un subconjunto respuestas generadas por la herramienta, comparando de forma directa la pregunta formulada, el contexto recuperado, la respuesta producida por el sistema y la evaluación asignada por el modelo de lenguaje utilizado como juez.

En primer lugar, se verificaron cinco casos de prueba relacionados con consultas de información contextual asociadas a distintos puntos de agua. Para cada uno de estos casos, se comparó la evaluación asignada por el enfoque *LLM-as-a-judge* con el criterio humano del evaluador, quien cuenta con conocimiento previo de la respuesta de referencia definida a partir de la información disponible en las bases de datos del sistema. La Tabla 4.7 presenta los cinco casos de prueba analizados, incluyendo la pregunta formulada, el contexto recuperado por el sistema, la respuesta generada por la herramienta, la respuesta de referencia, las métricas obtenidas mediante la evaluación automática y la retroalimentación del evaluador.

Pregunta	Contexto recuperado	Respuesta generada por la herramienta	Respuesta de referencia (*ground truth*)	Faithfulness	Answer Relevancy	Answer Similarity	Evaluación humana
In which village is Waterpoint Dinamo located?	Waterpoint Diingamoo Haro Dingamo is situated in Dingamo village, Kancharo Kebele, Dillo Woreda, Borana Zone of Oromia Region, Southern Ethiopia. The waterpoint is situated at 4.4736319 North latitude and 37.7130104 East longitude.	Waterpoint Dinamo is located in Dingamo village, Kancharo Kebele, Dillo Woreda, Borana Zone of Oromia Region, Southern Ethiopia.	Haro Dingamo is situated in Dingamo village.	1.00	0.89	0.51	La respuesta generada se considera válida y está sustentada por el contexto recuperado. La menor similitud se explica por diferencias de extensión entre la respuesta generada y la respuesta de referencia, sin evidencia de alucinación.
What is the total number of households in Waterpoint Bakke?	The total number of households using waterpoint Haro Bake waterpoint was 8,893, among which 5,325 were male-headed and 3,568 were female-headed.	The total number of households in Waterpoint Bakke is 8,893.	The total number of households using waterpoint Haro Bake is 8,893.	1.00	1.00	0.78	La respuesta es correcta y se encuentra respaldada por el contexto
What are the purposes of the Diimtuu waterpoint?	Waterpoint Diimtuu-Jaarsoo Xuyyee Haro Dimitu is used for multiple purposes such as drinking, domestic uses other than drinking, livestock watering and wildlife.	The purposes of the Diimtuu waterpoint include drinking, domestic use, livestock watering and wildlife.	Haro Dimitu is used for drinking, domestic uses, livestock watering and wildlife.	1.00	0.91	0.79	La respuesta se encuentra respaldada por el contexto recuperado y se considera correcta para la pregunta formulada.
What are the main challenges facing waterpoint Diimtuu?	The main challenges to the waterpoint management include poor coordination skills, limited water management capacity, poor support from different organizations, unwise use of resources and natural disasters such as drought and flood.	The main challenges include poor coordination, limited management capacity, lack of institutional support and natural disasters.	Challenges include poor coordination, limited management capacity, lack of support and droughts or floods.	0.86	1.00	0.81	La respuesta generada es consistente con el contexto recuperado y se considera correctamente sustentada.
Which population uses the Jiloo Dhokkichaa water point during the dry season?	The total population using the waterpoint during the dry season was 23,567 individuals, of which 11,589 (49 %) were female and 11,978 (51 %) were male.	The population using the waterpoint during the dry season is 23,567 people, with 49 % female and 51 % male.	The total population using the waterpoint during the dry season is 23,567 individuals.	1.00	0.95	0.77	La respuesta generada es consistente con el contexto recuperado y se considera correctamente sustentada.

Tabla 4.7: Casos seleccionados para validación humana y evaluación mediante *LLM-as-a-judge*

A continuación, se presenta una validación cualitativa específica para consultas orientadas al estado del punto de agua. Este tipo de consultas activa un flujo de procesamiento distinto, centrado en la identificación del estado del punto de agua y en la generación de recomendaciones asociadas.

La validación se realizó sobre un conjunto representativo de casos, en los cuales se contrastó la evaluación automática generada mediante el enfoque *LLM-as-a-judge* con el criterio humano del evaluador. Este análisis permitió verificar si las respuestas generadas por la herramienta reflejan de manera fiel el estado registrado. La Tabla 4.8 muestra los casos seleccionados para esta validación, incluyendo la pregunta formulada, el contexto recuperado por el sistema, la respuesta generada por la herramienta, la respuesta de referencia, las métricas obtenidas automáticamente y la evaluación cualitativa realizada por el evaluador humano.

Pregunta	**Contexto recuperado**	**Respuesta generada por la herramienta**	**Respuesta de referencia (*ground truth*)**	**Faithfulness**	**Answer Relevancy**	**Answer Similarity**	**Evaluación humana**
Give me the status of the waterpoint Dire	State: NEAR_DRY. Advisory: Pond is in near dry condition; water can be used for drinking after proper treatment.	The waterpoint Dire is in a near dry condition. It is recommended to use the pond water for drinking after proper treatment.	The waterpoint Dire is in a near dry condition. It is recommended to use the pond water for drinking after proper treatment.	0.67	0.69	1.00	La respuesta generada coincide con el estado y la recomendación presentes en el contexto recuperado. Se considera correctamente sustentada, aunque la fidelidad asignada por el evaluador automático resulta conservadora.
Status in the waterpoint Mudi Sororo	State: SEASONAL_DRY. Advisory: Pond is seasonally dry; please refer to ponds with available water.	The waterpoint Mudi Sororo is currently in a seasonal dry state, please text "ponds" to get ponds that have water.	The waterpoint Mudi Sororo is currently in a seasonal dry state, please text "ponds" to get ponds that have water.	0.67	0.86	0.92	La respuesta refleja correctamente el estado del punto de agua y la recomendación asociada. Se considera consistente con el contexto recuperado.
Status of the waterpoint Muyyattee	State: SEASONAL_DRY. Advisory: Pond is seasonally dry; please refer to ponds with available water.	The waterpoint Muyyattee is currently in a seasonal dry state, please text "ponds" to get ponds that have water.	The waterpoint Muyyattee is currently in a seasonal dry state, please text "ponds" to get ponds that have water.	0.67	0.94	0.91	La respuesta generada es correcta y está alineada con la información de estado presente en el contexto recuperado.
Please give the current status of the waterpoint Boru Wario	State: GOOD. Advisory: Pond is in good condition; proper management is recommended.	The waterpoint Boru Wario is in GOOD status. Proper management of water use for other purposes is recommended.	The waterpoint Boru Wario is in GOOD status. Proper management of water use for other purposes is recommended.	0.50	0.93	1.00	La respuesta coincide plenamente con el estado y la recomendación proporcionados por el contexto. El valor reducido de fidelidad indica una subestimación por parte del evaluador automático.
Actual status of the waterpoint Arghanne	State: NEAR_DRY. Advisory: Pond is in near dry condition; water can be used after proper treatment.	The waterpoint Arghanne is currently in a near dry condition. It is recommended to use the pond water for drinking after proper treatment.	The waterpoint Arghanne is currently in a near dry condition. It is recommended to use the pond water for drinking after proper treatment.	0.67	0.82	1.00	La respuesta generada está completamente sustentada por el contexto recuperado. La discrepancia en la métrica de fidelidad sugiere una limitación del enfoque *LLM-as-a-judge* en este caso.

Tabla 4.8: Casos seleccionados para validación humana en consultas de estado del punto de agua

Finalmente, se realizó una validación humana específica para las consultas orientadas a la recuperación de variables climáticas asociadas a los puntos de agua. Este tipo de consultas activa un flujo de acceso directo a datos estructurados, en el cual el sistema recupera valores numéricos específicos como precipitación, profundidad del agua, profundidad escalada y evapotranspiracion, en fechas determinadas.

La Tabla 4.9 muestra los casos seleccionados para esta validación, incluyendo las preguntas formuladas, el contexto recuperado por la herramienta, las respuestas generadas, las respuestas de referencia y las métricas de evaluación automática, junto con la valoración cualitativa realizada por el evaluador humano.

Pregunta	Contexto recuperado	Respuesta generada por la herramienta	Respuesta de referencia (*ground truth*)	Faithfulness	Answer Relevancy	Answer Similarity	Evaluación humana
Can you tell me the last rain record registered for the waterpoint Dire?	Params: waterpoint_id=..., waterpoint_name=Dire, variable=rain, date=current. Record: date=2024-10-15, rain=2.0 mm.	The last rain record for the waterpoint Dire is 2.0 mm on 2024-10-15.	The last rain record for the waterpoint Dire is 2.0 mm on 2024-10-15.	1.00	0.95	0.96	La respuesta coincide plenamente con el valor y la fecha registrados en el contexto recuperado, por lo que se considera correcta y adecuadamente sustentada.
Give me the scaled depth register on January 23, 2022 in Burra waterpoint	Params: waterpoint_id=..., waterpoint_name=Burra Gulanta, variable=scaled_depth, date=2022-01-23. Record: scaled_depth=96.4983.	The scaled depth register for Burra Gulanta on January 23, 2022, is 96.50 units.	The scaled depth register for Burra Gulanta on January 23, 2022, is 96.50.	1.00	0.81	0.98	La respuesta es correcta y está respaldada por el registro numérico recuperado; la diferencia se debe al redondeo del valor.
I need to know the record of all recent climatic variables for the Dinle waterpoint	Params: waterpoint_id=..., waterpoint_name=Dinle, variable=all, date=current. Record: depth=0.833666, evapotranspiration=4.25039, rain=0.0, scaled_depth=41.6833.	The recent climatic variables for the Dinle waterpoint are Depth: 0.833666 meters, Evapotranspiration (evp): 4.25039 mm, Rain: 0.0 mm and Scaled Depth: 41.6833.	The recent climatic variables for the Dinle waterpoint are Depth: 0.833666 meters, Evapotranspiration: 4.25039, 0.0 mm, Scaled Depth: 41.6833..	0.67	0.89	0.95	La respuesta se encuentra correctamente sustentada por el contexto recuperado; la agregación de múltiples variables explica la reducción en la fidelidad asignada.
Give me the last recorded depth for the Guratti waterpoint, just the number, absolutely nothing else.	Params: waterpoint_id=..., waterpoint_name=Gurratti, variable=depth, date=current. Record: depth=0.0.	0.0	0.0	1.00	1.00	0.72	La respuesta cumple estrictamente la instrucción de formato y coincide con el valor almacenado en el contexto recuperado.
Give the scaled depth, rain and depth of the waterpoint Ketala on the day of the presidential elections in Colombia 2022 (first round).	Params: waterpoint_id=..., waterpoint_name=Ketala, variable=all, date=2022-05-29. Record: depth=0.0, evapotranspiration=4.23437, rain=0.243647, scaled_depth=0.0.	On May 29, 2022, at the waterpoint Ketala, the values are as follows: scaled depth is 0.0, rain is 0.243647 mm, and depth is 0.0 m.	On May 29, 2022, at the waterpoint Ketala, the values are scaled depth 0.0, rain 0.243647 mm, and depth 0.0 m.	1.00	0.45	0.98	La respuesta generada está completamente respaldada por el contexto recuperado. El valor reducido de relevancia se asocia a la complejidad semántica de la consulta, que incluye una referencia temporal externa al dominio.

Tabla 4.9: Casos seleccionados para validación humana en consultas de variables climáticas

A partir de la revisión manual de las respuestas generadas para el conjunto reducido de casos de prueba seleccionado, se observo que la mayoría de las respuestas evaluadas se encuentran correctamente sustentadas por la información recuperada y son coherentes. Este análisis permitió identificar casos en los que la evaluación automatizada penaliza un poco las respuestas correctas debido a diferencias en la forma o extensión de la respuesta. En este sentido, la validación humana complementa la evaluación automática y contribuye a una interpretación más precisa del desempeño real de la herramienta. No obstante, debe tenerse en cuenta que el subconjunto de casos de prueba sometido a validación humana es reducido. En consecuencia, los resultados obtenidos a partir de estas quince respuestas no pueden considerarse una representación general del desempeño global de la herramienta. Su propósito es complementar la evaluación automatizada, aportando evidencia cualitativa sobre el comportamiento del sistema en escenarios específicos, más que establecer conclusiones definitivas.

4.1.6. Riesgos y limitaciones de la evaluación

A pesar de que la evaluación realizada proporciona una primera aproximación al desempeño de la herramienta, existen varios riesgos y limitaciones que deben ser considerados al interpretar los resultados obtenidos.

En primer lugar, el tamaño del conjunto de datos utilizado para la evaluación es relativamente

reducido, con un total de 40 ejemplos. Esta cantidad puede no ser suficiente para capturar la diversidad completa de consultas que podrían presentarse en un entorno real, lo que limita la capacidad de generalizar los resultados a un conjunto más amplio de preguntas.

En segundo lugar, existe un posible sesgo en la formulación de las preguntas, dado que estas fueron definidas por el desarrollador del sistema, quien posee conocimiento previo sobre su funcionamiento y comportamiento. Este hecho puede haber influido, consciente o inconscientemente, en la selección de preguntas que favorecen el desempeño de la herramienta.

Adicionalmente, la evaluación se basó en métricas automáticas, incluyendo enfoques del tipo *LLM-as-a-judge*, los cuales, aunque permiten escalar el proceso de evaluación, introducen una dependencia en el comportamiento del modelo evaluador y pueden reflejar sus propios sesgos o limitaciones.

Finalmente, los resultados obtenidos corresponden a una configuración específica del sistema (modelo utilizado, temperatura del modelo, parámetros de recuperación y tamaño de contexto), por lo que cambios en estas variables podrían afectar el desempeño observado.

Estas limitaciones deben tenerse en cuenta al interpretar los resultados y motivan la realización de evaluaciones futuras con conjuntos de datos más amplios, mayor diversidad de preguntas y, preferiblemente, con participación de evaluadores humanos externos.

4.2. Resumen del capítulo

En este capítulo se presentó la evaluación de la herramienta desarrollada, orientada a analizar el desempeño del sistema frente a distintos tipos de consultas asociadas a puntos de agua. Se definió un conjunto de casos de prueba representativos de los flujos de procesamiento del sistema y se describió la metodología empleada para su evaluación.

La evaluación se llevó a cabo mediante un enfoque automatizado basado en métricas para sistemas de recuperación aumentada con generación, lo que permitió caracterizar el comportamiento de la herramienta en términos de fidelidad, relevancia y similitud de las respuestas generadas. Adicionalmente, se incorporó un proceso de validación humana sobre un subconjunto de casos de prueba, con el fin de contrastar los resultados obtenidos automáticamente y evidenciar las limitaciones a la evaluación basada exclusivamente en modelos de lenguaje como juez.

En conjunto, los resultados presentados en este capítulo permiten evaluar de forma integral el desempeño de la herramienta y sientan las bases para la discusión de sus alcances, limitaciones y oportunidades de mejora, las cuales se abordan en los capítulos posteriores.

Capítulo 5

Conclusiones

5.1. Conclusiones

El objetivo de este trabajo fue diseñar, desarrollar y evaluar una herramienta capaz de reconocer consultas en lenguaje natural y recuperar información desde dos fuentes de información para generar respuestas contextualizadas sobre puntos de agua en Etiopía, corriendo en la interfaz conversacional de Telegram. Este objetivo se cumplió mediante la definición de estrategias para el acceso, almacenamiento, actualización y vectorización de los datos asociados a cada punto de agua, la construcción de una arquitectura basada en Recuperación aumentada con generación (RAG) usando el modelo C4, dividiendo esta en contexto del sistema, contenedores y componentes, y la implementación de dicha arquitectura en una herramienta que usa como interfaz conversacional la red social Telegram. La herramienta desarrollada permitió integrar dos modelos de lenguaje con un sistema de recuperación semántica de información, facilitando el acceso a datos a través de consultas en lenguaje natural. Adicionalmente, se realizó una evaluación sistemática del desempeño utilizando el framework RAGAS, específicamente la métrica de fidelidad (Faithfulness), lo que permitió verificar que las respuestas generadas se encontraban mayormente respaldadas por el contexto recuperado y superaban el umbral mínimo establecido de 0.70. En conjunto, los resultados evidencian que los objetivos planteados fueron alcanzados y que la herramienta constituye una solución para apoyar la consulta de información de puntos de agua y su información climática en Etiopía.

5.2. Relación con la literatura existente

Los resultados obtenidos en este trabajo se alinean y, al mismo tiempo, extienden los enfoques propuestos en trabajos previos relacionados con el uso de inteligencia artificial para el acceso a información especializada mediante interfaces conversacionales.

En comparación con *Demeter chatbot* (Sotelo, 2021), que utiliza técnicas de procesamiento de lenguaje natural basadas en BERT y respuestas predefinidas enriquecidas con información recuperada desde una API, la herramienta desarrollada en este proyecto incorpora un modelo generativo basado en LLM junto con una arquitectura de recuperación aumentada por generación (RAG). Esto permite producir respuestas más flexibles y adaptadas al contexto de cada consulta, superando la limitación de respuestas estrictamente predefinidas identificada en dicho trabajo.

Respecto al chatbot propuesto por (Vakayil et al., 2024), orientado al apoyo a víctimas de acoso sexual y basado en Llama-2 con arquitectura RAG, este trabajo comparte el uso de técnicas modernas de recuperación de información y generación aumentada. No obstante, mientras dicho sistema se encuentra limitado a un entorno local, la herramienta desarrollada en este proyecto fue diseñada para ser desplegada como una interfaz conversacional accesible a través del canal de mensajería Telegram, lo que amplía su alcance potencial hacia usuarios finales como agropastores y extensionistas.

En relación con el trabajo de (Quidwai and Lagana, 2024) en el contexto del mieloma múltiple, ambos sistemas emplean arquitecturas RAG para facilitar el acceso a información especializada. Sin embargo, mientras ese trabajo se enfoca en un entorno clínico restringido y utiliza infraestructuras específicas como Amazon Kendra y OpenSearch, la herramienta propuesta en este proyecto prioriza que la herramienta este en un entorno abierto para el uso no solo del publico objetivo sino también de cualquier usuario que desee y necesite la información.

Finalmente, frente al sistema *Waterpoints Monitoring* (Alemayehu et al., 2024b), que proporciona información técnica detallada sobre puntos de agua mediante interfaces visuales, la principal contribución de este trabajo consiste en habilitar el acceso a dicha información a través de lenguaje natural y canales conversacionales. Esto permite reducir la barrera técnica para el acceso a los datos y facilita su uso por parte de usuarios no especializados.

En conjunto, estos resultados muestran que la herramienta desarrollada complementa y amplía los enfoques existentes, integrando técnicas de recuperación y generación de lenguaje con un enfoque centrado en la disponibilidad a través de canales conversaciones, lo que constituye una contribución amplia frente a las soluciones reportadas en la literatura.

5.3. Limitaciones de la investigación

Una de las principales limitaciones de este trabajo está relacionada con la disponibilidad y riqueza de los datos descriptivos asociados a cada punto de agua. En particular, la cantidad de información textual disponible en el idioma inglés para describir cada punto de agua era reducida, lo que implicó que la base de datos vectorizada contuviera representaciones limitadas del contexto real de cada punto de agua. Como consecuencia, la herramienta disponía de un conjunto restringido de información, lo que afectó tanto la diversidad de las consultas que podían formularse como la profundidad de las respuestas que la herramienta podía generar. Esta limitación también influyó en el diseño del conjunto de preguntas utilizado para la evaluación, el cual se vio necesariamente reducido por la escasez de información disponible en la base de datos.

Por otra parte, la segunda limitación identificada durante el desarrollo del trabajo está relacionada con la disponibilidad de las fuentes de datos externas. En la fase inicial de formulación de la propuesta se contempló la inclusión de una funcionalidad orientada a la generación de información pronósticos climáticos para cada punto de agua. Esta propuesta se definió en un contexto en el que

la fuente de información utilizada para alimentar los datos de pronóstico se encontraba operativa.

Pero, durante el desarrollo del presente trabajo dicha fuente de datos dejó de funcionar, lo que implicó la disponibilidad únicamente de información desactualizada. Dado que el uso de datos desactualizados contradice el principio fundamental de un pronóstico, esta funcionalidad no pudo ser implementada en la herramienta desarrollada. En consecuencia, el alcance del sistema se limitó a información histórica y actual disponible, excluyendo la generación de pronósticos como parte de los resultados del presente trabajo.

Por ultimo, se reconoció como limitación del presente trabajo, la ausencia de una evaluación comparativa de tipo A/B entre un modelo base sin mecanismos de recuperación y la versión de la herramienta con recuperación aumentada con generación (RAG). Esta decisión se fundamenta en la naturaleza de las consultas abordadas, las cuales dependen en su totalidad de información específica del dominio, almacenada en fuentes de datos internas y no disponible en el conocimiento del ningún modelo de lenguaje.

En este contexto, una evaluación sin recuperación habría conducido a respuestas incorrectas en todos los casos, impidiendo una comparación significativa del desempeño del sistema. Por esta razón, la evaluación se centró en analizar la herramienta en su configuración prevista de uso, donde es posible medir de forma objetiva métricas asociadas a la fidelidad, relevancia y similitud de las respuestas generadas.

5.4. Trabajos futuros

Como líneas de trabajo futuro, se plantea en primer lugar la ampliación de la herramienta hacia otros canales e interfaces conversacionales, tales como plataformas de mensajería adicionales (por ejemplo, Facebook Messenger y WhatsApp) y aplicaciones web, con el fin de incrementar su alcance y facilitar el acceso por parte de distintos tipos de usuarios.

En segundo lugar, se propone extender el soporte lingüístico de la herramienta a otros idiomas locales utilizados en Etiopía, particularmente al amárico y al oromo, lo que permitiría mejorar la accesibilidad del sistema y facilitar su adopción por parte de comunidades que no utilizan el inglés como idioma principal.

Adicionalmente, se considera fundamental incorporar un enfoque formal de diseño centrado en el usuario. Si bien el sistema fue diseñado teniendo en cuenta el perfil general de los usuarios objetivo (agropastores, extensionistas y tomadores de decisiones), no se realizó una validación estructurada con usuarios finales durante la presente fase del proyecto.

Como trabajo futuro se propone realizar sesiones de evaluación con usuarios reales que permitan analizar aspectos como usabilidad, claridad de las respuestas generadas, facilidad de interacción en

Telegram, tiempo de resolución de consultas y nivel de comprensión del lenguaje empleado.

La incorporación de principios de diseño centrado en el usuario permitiría ajustar la herramienta y mejorar la experiencia general de uso del sistema en contextos reales.

Adicionalmente, como línea de mejora futura, se plantea incorporar un mecanismo de georreferenciación que permita identificar automáticamente la ubicación aproximada del usuario y priorizar la entrega de información asociada a los puntos de agua más cercanos.

Asimismo, se propone la incorporación de un menú interactivo dentro de la interfaz de Telegram que incluya preguntas frecuentes o consultas más relevantes, siguiendo un enfoque tipo Pareto. Esto permitiría guiar a los usuarios hacia las consultas más comunes y reducir la ambigüedad en las preguntas, mejorando la experiencia de uso y la eficiencia del sistema.

5.5. Lecciones aprendidas

Durante el desarrollo del presente proyecto se obtuvieron diversas lecciones aprendidas tanto a nivel técnico como metodológico. En el ámbito técnico, se adquirió experiencia en la vectorización de datos textuales mediante el uso de modelos de *embeddings*, así como en el diseño e implementación de mecanismos de búsqueda semántica sobre bases de datos vectorizadas. Estos componentes fueron una parte muy importantes para el desarrollo del proyecto, puesto que era el núcleo de la herramienta desarrollada.

Asimismo, se profundizó en el uso de modelos de lenguaje de gran tamaño (LLM), tanto en entornos locales como en servicios en la nube, así como en su orquestación dentro de flujos de procesamiento que combinan recuperación de información y generación de respuestas en lenguaje natural. Este aprendizaje permitió comprender las diferencias, de rendimiento y de configuración entre ambos enfoques.

Desde una perspectiva metodológica, el proyecto permitió explorar y aplicar técnicas de evaluación automatizada para sistemas RAG, incluyendo el uso de métricas orientadas a medir fidelidad, relevancia y similitud de las respuestas generadas. Adicionalmente, se evidenció la importancia de complementar estas métricas con validación humana, especialmente en dominios donde la corrección factual depende de datos estructurados y reglas de negocio específicas.

Finalmente, a nivel personal y profesional, el desarrollo de este proyecto permitió fortalecer habilidades del mi rol de ingeniero de software, tales como el diseño de arquitectura de software, la integración de múltiples fuentes de datos y la toma de decisiones técnicas basadas en restricciones reales de un sistema y dominio. Asimismo, adquirí experiencia en la gestión de sistemas que usan modelos de lenguaje, incluyendo el manejo de prompts, parámetros de los modelos como temperatura, la identificación de limitaciones de estos modelos y la necesidad de validación del contenido que generan. Estas lecciones resultan importantes para el desarrollo profesional de mi carrera como

profesional.

Bibliografía

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alemayehu, S. (2022). A real-time livestock water source monitoring and risk management system project in ethiopia. https://alliancebioversityciat.org/stories/real-time-livestock-water-source-monitoring-and-risk-management-system-project-ethiopia.

Alemayehu, S. and Workeneh, S. (2023). Borana pastoralist struggling to survive under the recurrent drought in ethiopia. https://alliancebioversityciat.org/stories/borana-pastoralist-struggling-survive-under-recurrent-drought-ethiopia.

Alemayehu, S., Workeneh, S., Gebre, L., Tegegne, G., Tesfaye, L., Abdulhamid, N., Yaregal, Y., and Terefe, T. (2024a). Advancing anticipatory actions: The regional launch of rangeland and water monitoring system in ethiopia. https://alliancebioversityciat.org/stories/launching-user-centered-rangeland-water-monitoring-system-early-warning-ethiopia.

Alemayehu, S., Workeneh, S., Tegegn, G., Terefe, T., Tesfaye, L., and Gebre, L. (2024b). New digital system helps ethiopia's pastoralist communities to prepare for the unpredictable. https://alliancebioversityciat.org/stories/new-digital-system-helps-ethiopias-pastoralist-communities-prepare-unpredictable.

Amazon Web Services (2024). ¿qué es un modelo de lenguaje grande (llm)? Recuperado el 7 de abril de 2025.

Amazon Web Services (2025). ¿qué es langchain? Recuperado el 15 de enero de 2025.

Avthar Sewrathan (2023). Postgresql como base de datos vectorial. Recuperado el 15 de enero de 2025.

Bahru, Y. (2022). Why telegram is so popular in ethiopia? Addis Insight, accessed April 6, 2025.

Bass, L., Clements, P., and Kazman, R. (2021). *Software architecture in practice*. Addison-Wesley Professional.

Corredera, J. C. (2023). Inteligencia artificial generativa. In *Anales de la Real academia de Doctores*, volume 8, pages 475–489.

Es, S., James, J., Espinosa Anke, L., and Schockaert, S. (2024). RAGAs: Automated evaluation of retrieval augmented generation. In Aletras, N. and De Clercq, O., editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Esden Business School (2025). ¿Qué es Claude y para qué sirve? Consultado el 30 de enero de 2025.

García-Peñalvo, F. J. (2023). Discusión abierta sobre beneficios, riesgos y retos de la inteligencia artificial generativa.

González Alcaide, G. (2024). Inteligencia artificial generativa: Un contexto disruptivo en el acceso a la información. *Infonomy*, 2(1).

James, H. (2025). Deepseek local: How to self-host deepseek (privacy and control). Blog Linux.

Kukreja, S., Kumar, T., Bharate, V., Purohit, A., Dasgupta, A., and Guha, D. (2023). Vector databases and vector embeddings-review. In *2023 International Workshop on Artificial Intelligence and Image Processing (IWAIIP)*, pages 231–236.

Meta (2024). Presentamos llama 3.1: nuestro modelo de lenguaje a gran escala más capaz hasta la fecha. Recuperado el 15 de enero de 2025.

Mistral. AI (2025). Evaluación de rag con llm como juez. Recuperado el 15 de enero de 2025.

Mollá, V. (2024). ¿qué es rag? una gran arquitectura en llms.

Nassiri, K. and Akhloufi, M. (2023). Transformer models used for text-based question answering systems. *Applied Intelligence*, 53(9):10602–10635.

Navarro, S. (2024). ¿qué son los modelos preentrenados? Último acceso: 18 de febrero de 2026.

Nuñez, R. (2008). Agro-pastores de la región somalí de etiopía:"proteger el ganado es un seguro de vida para estas familias". *REDVET. Revista Electrónica de Veterinaria*, 9(10):1–6.

OpenAI (2024a). Gpt-4o mini: avanzando en inteligencia rentable. Recuperado el 15 de enero de 2025.

OpenAI (2024b). text-embedding-3-small. Recuperado el 15 de enero de 2025.

Osornio, A. (2025). DeepSeek: Todo lo que debes saber sobre el equivalente chino de ChatGPT. Consultado el 29 de enero de 2025.

Quidwai, M. A. and Lagana, A. (2024). A rag chatbot for precision medicine of multiple myeloma. *medRxiv*, pages 2024–03.

Rivera, W. M. and Qamar, M. K. (2003). *Agricultural extension, rural development and the food security challenge*. Food and Agriculture Organization of the United Nations Rome.

Sarkar, D. S. (2024). Preferences, choice and constrains of social media platforms in ethiopia. *Pakistan Journal of Media Sciences*, 5(1):35–41.

Sofia Hansen (2025). Qué es n8n y cómo empezar a automatizar procesos sin saber programar. Recuperado el 15 de enero de 2025.

Sotelo, S. (2021). Demeter: Un sistema chatbot de información agroclimática para colombia. Master's thesis, Universitat Oberta de Catalunya.

Telegram (2024). Bots: An introduction for developers. https://core.telegram.org/bots. Accedido el 19 de abril de 2025.

Vakayil, S., Juliet, D. S., J, A., and Vakayil, S. (2024). Rag-based llm chatbot using llama-2. In *2024 7th International Conference on Devices, Circuits and Systems (ICDCS)*, pages 1–5.

Velasco-Elizondo, P. (2015). *Arquitectura de Software: Conceptos y Ciclo de Desarrollo.*

Waterpoint Data Exchange (2025). Waterpoint data exchange. https://tools.waterpointdata.org/?all_waterpoints=true&any_waterpoints=true&show_population_density=false&show_landcover=false&show_adm_borders=false&show_point_counts=false&adman_view=%22unserved%22&show_adman_pies=true&show_adman_labels=true&mode=%22basic%22&adman_level=%22best%22&nc_limit=0&show_adm=false&sort_by=%22would_gain_access%20desc%22&country_name=%22Ethiopia%22&bounds=%5B%5B32.99180000000007,3.4066700000000765%5D,%5B47.988240000000076,14.845476907000034%5D%5D. Accessed: 2025-03-15.

Wu, J. (2025). The rise of deepseek: technology calls for the "catfish effect". *Journal of Thoracic Disease*, 17(2):1106.

Zeichick, A. (2023). ¿Qué es la generación aumentada de recuperación (RAG)? Consultado el 19 de septiembre de 2023.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena.