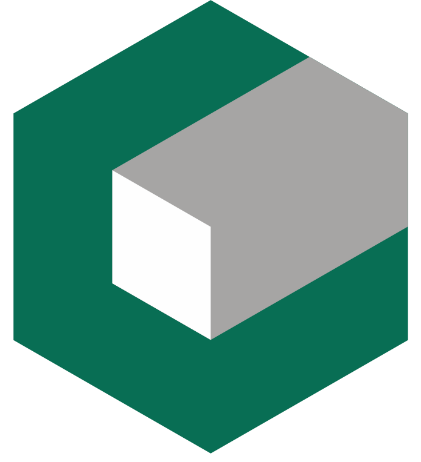
Universidad Tecnológica de La Habana

“ José Antonio Echeverría “

Facultad de Ingeniería Informática



**Asistente virtual de tipo chatbot para el sistema de la biblioteca de la Universidad Tecnológica José Antonio Echeverría**

Autor:

Alejandro Estrada Suárez

Tutora:

Dra. C. Nayma Cepero Pérez

La Habana, Cuba

Enero del 2025

**Resumen**

La Inteligencia Artificial (IA), una rama de la ciencia de la computación dedicada a la automatización de la conducta inteligente, ha transformado significativamente la manera en que las instituciones procesan y acceden a la información, ofreciendo soluciones innovadoras. Dentro de sus áreas más destacadas se encuentra el Procesamiento del Lenguaje Natural (PLN), que permite a las máquinas comprender y generar texto de manera coherente utilizando el lenguaje humano. Los modelos de lenguaje de grandes (LLM) han jugado un papel fundamental en esta revolución, ya proporcionando capacidades avanzadas para interpretar y responder a consultas diversas. En este contexto, la técnica de Generación Aumentada por Recuperación (RAG) combina a los LLM con bases de conocimiento para generar respuestas precisas y relevantes basadas en un dominio específico. Estos avances junto a los agentes inteligentes, definidos como componentes de software autónomos capaces de percibir su entorno, razonar y actuar para cumplir objetivos específicos, han dado paso a la creación de asistentes virtuales de tipo chatbots, que integran este tipo de agentes, permiten a los usuarios interactuar con sistemas informáticos de manera natural, respondiendo preguntas y proporcionando asistencia de forma automatizada. El presente trabajo desarrolla un asistente virtual de tipo chatbot diseñado específicamente para la biblioteca de la Universidad Tecnológica de La Habana José Antonio Echeverría (CUJAE). Este chatbot integra un agente inteligente capaz de responder preguntas relacionadas con la consulta de materiales y de servicios bibliotecarios. De esta manera, se representa una experiencia más accesible, eficiente y personalizada para los usuarios de la biblioteca.

Palabras clave: Inteligencia Artificial, Procesamiento del Lenguaje Natural, modelos de lenguaje grandes, Generación Aumentada por Recuperación, agentes inteligentes, asistentes virtuales, chatbot, biblioteca

**Abstract**

Artificial Intelligence (AI), a branch of computer science dedicated to the automation of intelligent behavior, has significantly transformed the way institutions process and access information, offering innovative solutions. Among its most prominent areas is Natural Language Processing (NLP), which allows machines to understand and generate text in a coherent manner using human language. Large language models (LLM) have played a fundamental role in this revolution, already providing advanced capabilities to interpret and respond to diverse queries. In this context, the Retrieval Augmented Generation (RAG) technique combines LLM with knowledge bases to generate precise and relevant responses based on a specific domain. These advances, together with intelligent agents, defined as autonomous software components capable of perceiving their environment, reasoning and acting to meet specific objectives, have given way to the creation of chatbot-type virtual assistants, which integrate this type of agents, allowing users to interact with computer systems in a natural way, answering questions and providing assistance in an automated way. This paper presents a chatbot-type virtual assistant designed specifically for the library of the José Antonio Echeverría Technological University of Havana (CUJAE). This chatbot integrates an intelligent agent capable of answering questions related to the consultation of library materials and services. In this way, a more accessible, efficient and personalized experience is represented for library users.

Keywords: Artificial Intelligence, Natural Language Processing, large language models, Retrieval Augmented Generation, intelligent agents, virtual assistants, chatbot, library

**Índice General**

[Introducción 1](#_Toc187922081)

[Capítulo 1: Fundamentos Teóricos 6](#_Toc187922082)

[1.1. Introducción 6](#_Toc187922083)

[1.2. Asistentes virtuales de tipo chatbot 6](#_Toc187922084)

[1.2.1. Características de los chatbots 9](#_Toc187922085)

[1.2.2. Clasificación de los chatbots 10](#_Toc187922086)

[1.2.3. Arquiteturas de chatbots 11](#_Toc187922087)

[1.3. Procesamiento del Lenguaje Natural 12](#_Toc187922088)

[1.4. Modelo de Lenguaje Grande 14](#_Toc187922089)

[1.5. Generación Aumentada por Recuperación 16](#_Toc187922090)

[1.6. Agentes Inteligentes 17](#_Toc187922091)

[1.6.1. Características de los agentes inteligentes 18](#_Toc187922092)

[1.7. Chatbots en bibliotecas universitarias 19](#_Toc187922093)

[1.8. Tecnologías para el desarrollo de chatbots 21](#_Toc187922094)

[1.8.1. Dialogflow 22](#_Toc187922095)

[1.8.2. IBM Watson 22](#_Toc187922096)

[1.8.3. Amazon Lex 23](#_Toc187922097)

[1.8.4. Rasa 24](#_Toc187922098)

[1.8.5. Langchain 24](#_Toc187922099)

[1.9. Selección de la tecnología pra el desarrollo del chatbot 25](#_Toc187922100)

[1.9.1. Langchain 25](#_Toc187922101)

[1.9.2. Modelo de generación de embeddings de texto 26](#_Toc187922102)

[1.9.3. LLM 28](#_Toc187922103)

[1.9.4. Ollama 29](#_Toc187922104)

[1.9.5. Chroma DB 30](#_Toc187922105)

[1.9.6. Mongo DB 31](#_Toc187922106)

[1.9.7. FastApi 32](#_Toc187922107)

[1.10. Conclusiones parciales 32](#_Toc187922108)

[Capítulo 2: Solución Propuesta 34](#_Toc187922109)

[2.1. Introducción 34](#_Toc187922110)

[2.2. Captura de requisitos 34](#_Toc187922111)

[2.2.1. Requisitos funcionales 34](#_Toc187922112)

[2.2.2. Requisitos de calidad 35](#_Toc187922113)

[2.3. Casos de Uso del Sistema 36](#_Toc187922114)

[2.3.1. Actores del sistema 36](#_Toc187922115)

[2.3.2. Diagrama de Casos de Uso del Sistema 37](#_Toc187922116)

[2.3.3. Descripción de los Casos de Uso del Sistema 38](#_Toc187922117)

[2.4. Arquitectura de la solución propuesta 45](#_Toc187922118)

[2.4.1. Estilo arquitectónico: Llamada y Retorno 45](#_Toc187922119)

[2.4.1.1. Patrón arquitectónico: N - Capas 45](#_Toc187922120)

[2.4.1.2. Patrón arquitectónico: Cliente Servidor 47](#_Toc187922121)

[2.4.2. Estilo arquitectónico: Centrado en datos 48](#_Toc187922122)

[2.4.2.1. Patrón arquitectónico: Pizarra 49](#_Toc187922123)

[2.4.3. RAG con uso de agente 50](#_Toc187922124)

[2.5. Diseño e implementación 52](#_Toc187922125)

[2.5.1. Patrones de diseño 52](#_Toc187922126)

[2.5.2. Principios de diseño 56](#_Toc187922127)

[2.5.3. Aspectos de implementación 57](#_Toc187922128)

[2.6. Almacenamiento de Datos 62](#_Toc187922129)

[2.7. Modelo de despliegue 64](#_Toc187922130)

[2.8. Conclusiones parciales. 67](#_Toc187922131)

[Capítulo 3: Validación de la solución 68](#_Toc187922132)

[3.1. Introducción 68](#_Toc187922133)

[3.2. Pruebas funcionales 68](#_Toc187922134)

[3.2.1. Guiadas por casos de uso 68](#_Toc187922135)

[3.3. Conclusiones parciales 91](#_Toc187922136)

[Conclusiones generales 92](#_Toc187922137)

[Recomendaciones 93](#_Toc187922138)

[Referencias Bibliográficas 94](#_Toc187922139)

**Índice de Figuras**

[Figura 1: Massive Text Embedding Benchmark 27](#_Toc187948987)

[Figura 2: Comparación de Llama 3.1 8B con otros LLMs 29](#_Toc187948988)

[Figura 3: Diagrama de Casos de Uso del Sistema 37](#_Toc187948989)

[Figura 4: Representación del Patrón Arquitectónico N - Capas 46](#_Toc187948990)

[Figura 5: Representación del Patrón Arquitectónico Cliente Servidor 48](#_Toc187948991)

[Figura 6: Representación del Patrón Arquitectónico Pizarra 49](#_Toc187948992)

[Figura 7: Proceso del de RAG utilizando un agente 50](#_Toc187948993)

[Figura 8: Diagrama de Clases UML de las clases utilizadas de Langchain 55](#_Toc187948994)

[Figura 9: Patrón de diseño Singleton 56](#_Toc187948995)

[Figura 10: Implementación de herramientas para el agente 59](#_Toc187948996)

[Figura 11: Implementación de la plantilla de chat 60](#_Toc187948997)

[Figura 12: Implementación del agente 61](#_Toc187948998)

[Figura 13: Colección chat\_histories 64](#_Toc187948999)

[Figura 14: Interacción con el chatbot mediante una interfaz de usuario de prueba para pedir información de un libro 71](#_Toc187949000)

[Figura 15: Proceso realizado por el agente para dar respuesta a la consulta del usuario 72](#_Toc187949001)

**Índice de Tablas**

[Tabla 1: Descripción del caso de uso “Insertar información contextual de la biblioteca” 38](#_Toc187921383)

[Tabla 2: Descripción del caso de uso “Modificar información contextual de la biblioteca” 38](#_Toc187921384)

[Tabla 3: Descripción del caso de uso “Eliminar información contextual de la biblioteca” 39](#_Toc187921385)

[Tabla 4: Descripción del caso de uso “Listar información contextual de la biblioteca” 40](#_Toc187921386)

[Tabla 5: Descripción del caso de uso “Consultar información de libros” 40](#_Toc187921387)

[Tabla 6: Descripción del caso de uso “Solicitar recomendaciones de libros” 41](#_Toc187921388)

[Tabla 7: Descripción del caso de uso “Consultar servicios ofrecidos en la biblioteca” 42](#_Toc187921389)

[Tabla 8: Descripción del caso de uso “Realizar búsqueda semántica” 43](#_Toc187921390)

[Tabla 9: Descripción del caso de uso “Actualizar historial del chat” 43](#_Toc187921391)

[Tabla 10: Descripción del caso de uso “Eliminar historial del chat” 44](#_Toc187921392)

[Tabla 11: Caso de Prueba Consultar información  libros 68](#_Toc187921393)

[Tabla 12: Caso de Prueba Solicitar recomendaciones de libros 73](#_Toc187921394)

[Tabla 13: Caso de Prueba Consultar servicios de la biblioteca 77](#_Toc187921395)

[Tabla 14: Caso de Prueba Eliminar chat 82](#_Toc187921396)

[Tabla 15: Caso de Prueba Insertar información contextual 83](#_Toc187921397)

[Tabla 16: Caso de Prueba Modificar información contextual 85](#_Toc187921398)

[Tabla 17: Caso de Prueba Eliminar información contextual 87](#_Toc187921399)

[Tabla 18: Caso de Prueba Listar información contextual 89](#_Toc187921400)

# **Introducción**

La Inteligencia Artificial (IA) es considerada como un medio para el desarrollo de sistemas que piensen y actúen racionalmente [1] . Una de sus definiciones plantea:  “es la rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente” [2] . Actualmente, la IA se encuentra integrada en la optimización de procesos y tareas tanto en grandes organizaciones como en la vida cotidiana de las personas. A través de diversas técnicas, esta tecnología facilita múltiples aspectos de la vida diaria y permite realizar tareas que antes requerían intervención humana [3].

La IA ha revolucionado las formas tradicionales de gestión, ofreciendo mejoras significativas en la calidad de bienes y servicios [4]. En particular, la automatización del servicio al cliente ha facilitado la interacción con organizaciones, respondiendo a la creciente demanda de sistemas más rápidos y eficientes. Esta evolución se ha vuelto crucial tanto en el sector público como en el privado, impulsando la adopción de herramientas tecnológicas para optimizar la atención a ciudadanos y clientes. Es por ello por lo que se ha optado por desarrollar asistentes virtuales, principalmente de tipo chatbot, que permitan el acceso en tiempo real a información y apoyo referente a los bienes y servicios brindados por las organizaciones públicas y privadas [5].

La idea de los chatbots tiene su origen en los años cincuenta, cuando Alan Turing se preguntó si una máquina podría comunicarse de manera tan natural que los humanos no detectaran que no estaban hablando con otra persona. Esta cuestión inspiró la creación de ELIZA en 1966, considerado el primer chatbot capaz de simular una conversación con un terapeuta. Aunque las capacidades de ELIZA eran limitadas, marcó un punto de partida importante para futuros desarrollos en este campo. Con el tiempo, el avance de la inteligencia artificial dio lugar a asistentes virtuales más sofisticados, como Siri de Apple, Cortana de Microsoft, Google Now y Google Assistant, así como Alexa de Amazon [6] .

Un asistente virtual puede ser un agente conversacional desarrollado con inteligencia artificial, particularmente con procesamiento de lenguaje natural que interactúa con los usuarios respondiendo sus consultas/preguntas. En este contexto los chatbots de propósitos específicos sirven para resolver las cuestiones sobre un dominio específico, como una tienda, un museo, o una institución [7]. En movilidad han sido interesantes en la asistencia a compras [8], mejorar la productividad en las tiendas [9] y en la asistencia de personas mayores en entornos de hogar [10] por su capacidad para resolver problemas concretos en el lugar. En el ámbito de la salud son usados para diagnosticar a pacientes [11], detectar los síntomas del paciente en base a la conversación con el usuario [12], así como detectar una posible enfermedad, a partir de los síntomas que el paciente proporciona [13].

En los entornos educativos, la IA impulsa la creación de procesos de enseñanza y aprendizaje más flexibles y personalizados, ajustándose a las necesidades específicas de los participantes. Además, permite el desarrollo de entornos inteligentes de aprendizaje que destacan por su adaptabilidad, autonomía y flexibilidad [14]. El impacto de la IA en la educación se está volviendo cada vez más significativo, con aplicaciones que abarcan desde la analítica de datos hasta el reconocimiento facial para interpretar las expresiones de los estudiantes [15].En este contexto, los chatbots se posicionan como herramientas prometedoras al ofrecer interacción con los alumnos de forma similar a la de un profesor, proporcionando retroalimentación en tiempo real, resolviendo dudas y guiando a los estudiantes en actividades educativas adaptadas a sus necesidades [16].

Las bibliotecas durante la historia han desempeñado un papel fundamental en el desarrollo de la sociedad, actuando como entidad encargada de preservar el conocimiento a través de documentos bibliográficos impresos. Con el avance de las Tecnologías de la Información y la Comunicación (TIC), las bibliotecas institucionales enfrentan la necesidad de desarrollar herramientas de amplio alcance y adaptarse rápidamente a los cambios tecnológicos [17]. Esta evolución responde al hecho de que sus usuarios, en su mayoría, pertenecen a generaciones de nativos e inmigrantes digitales, quienes demandan servicios cada vez más modernos y eficientes para satisfacer sus necesidades informativas [17].

En las bibliotecas académicas se han venido implementando chatbots como una manera de optimizar el tiempo en respuesta a los usuarios en lo relacionado con los servicios que se ofrecen en estas unidades de información, un ejemplo de esta tecnología se puede constatar con la universidad de Oklahoma, el chatbot de esta biblioteca se llama Bizzy y colabora con la respuesta de preguntas frecuentes [18]. Otro ejemplo lo constituye la Universidad de La Laguna donde se desarrolló un asistente virtual para la gestión de servicios de su biblioteca [19].

La biblioteca de la Universidad Tecnológica José Antonio Echeverría (CUJAE) es un pilar esencial en el soporte académico para estudiantes, profesores e investigadores; sin embargo, el sistema de gestión actual presenta limitaciones importantes en la atención al usuario y el acceso a la información. Los estudiantes que buscan orientación o acceso a material específico encuentran dificultades que afectan tanto la eficiencia en la recuperación de información como su satisfacción general. Esta situación se complica para aquellos usuarios que no dominan los procesos de búsqueda en los catálogos o desconocen los recursos digitales disponibles, enfrentando barreras para acceder a materiales relevantes para su desarrollo académico. Además, el sistema carece de herramientas automatizadas de recomendación que permitan personalizar la experiencia del usuario sugiriendo libros o recursos en función de sus intereses, lo que limita el uso eficiente del fondo bibliográfico. Paralelamente, la constante necesidad de responder preguntas frecuentes sobre servicios como préstamos, renovaciones y horarios consume tiempo y recursos humanos en el ámbito bibliotecario.

Dada esta **situación problemática** surge como **problema de la investigación**: ¿Cómo desarrollar un asistente virtual de tipo chatbot que permita a los usuarios de la biblioteca de la Universidad Tecnológica José Antonio Echeverría  la retroalimentación en cuanto a servicios ofrecidos por la misma así como en la sugerencia de libros según sus intereses?

Se define como **objeto de estudio:** el desarrollo de un asistente virtual inteligente; centrado en el **campo de acción** que comprende: la búsqueda y recuperación de información sobre libros y servicios ofrecidos por la biblioteca universitaria mediante un chatbot.

Para darle solución a este problema se plantea como **objetivo general**: Desarrollar un asistente virtual de tipo chatbot que permita a los usuarios de la biblioteca de la Universidad Tecnológica José Antonio Echeverría la retroalimentación en cuanto a servicios ofrecidos por la misma así como en la sugerencia de libros según sus intereses.

Con el fin de lograr este objetivo general, se han definido los siguientes **objetivos específicos**, junto con las **tareas** correspondientes:

1. Estudiar acerca de los conceptos vinculados a asistentes virtuales.
   1. Investigar acerca de las características de los asistentes virtuales  de tipo chatbot.
   2. Investigar acerca de la clasificación de los asistentes virtuales  de tipo chatbot.
   3. Investigar acerca de la arquitectura de los asistentes virtuales de tipo chatbot.
   4. Investigar acerca de el uso de asistentes virtuales  de tipo chatbots en bibliotecas universitarias.
   5. Investigar acerca de tecnologías para el desarrollo de asistentes virtuales de tipo chatbot.
   6. Seleccionar tecnología para el desarrollo del asistente virtual de tipo chatbot.
2. Desarrollar módulo del asistente virtual de tipo chatbot para el sistema de la biblioteca de la Universidad Tecnológica José Antonio Echeverría.
   1. Definir los requisitos de software.
   2. Diseñar el asistente virtual de tipo chatbot.
   3. Implementar el asistente virtual de tipo chatbot.
   4. Implementar memoria del asistente virtual de tipo chatbot.
   5. Definir cómo interactúa el módulo del asistente virtual de tipo chatbot con otros módulos de la biblioteca.
3. Validar la solución propuesta.
   1. Diseñar los casos de pruebas que representen escenarios de interacción con el asistente virtual de tipo chatbot.
   2. Ejecutar los casos de prueba diseñados.
   3. Analizar los resultados para identificar el rendimiento y la eficacia de las respuestas del asistente virtual de tipo chatbot.

La evaluación de la investigación está determinada por la **conveniencia**: busca mejorar la interacción entre los usuarios y los servicios de la biblioteca, facilitando el acceso a la información y recursos disponibles, la **relevancia social:** beneficia a estudiantes, profesores e investigadores al mejorar el acceso a la información y recursos bibliográficos, el **valor práctico:** mejora la atención al usuario, reduce tiempos de espera y facilita la búsqueda de materiales y el **valor teórico:** contribuye a la literatura sobre asistentes virtuales de tipo chatbot en bibliotecas universitarias. Los hallazgos pueden abrir la puerta a futuras investigaciones sobre la efectividad de herramientas digitales en la educación.

El trabajo estará dividido en tres capítulos. El Capítulo 1: Fundamentos teóricos, donde se mostrarán los diferentes resultados encontrados acerca de los chatbots, su estructura y las principales tecnologías que permiten implementar un asistente. El Capítulo 2: Solución propuesta, que contendrá los artefactos de Ingeniería de Software que modelan la solución. Por último, el Capítulo 3: Validación de la solución, donde se realizarán las pruebas guiadas por casos de uso.

# **Capítulo 1: Fundamentos Teóricos**

## **Introducción**

Este capítulo se centra en los conceptos fundamentales, presentando el tema de los asistentes virtuales de tipo chatbot, se examinan sus características, clasificaciones así como las arquitecturas referentes a ellos. Se caracteriza el uso de este tipo de asistente virtual en bibliotecas universitarias y además se presentan distintas tecnologías existentes para el desarrollo de estos chatbots.

## **Asistentes virtuales de tipo chatbot**

Los asistentes virtuales son sistemas de software que utilizan IA para interactuar con los usuarios de manera conversacional, proporcionando respuestas a preguntas, ayudando en la resolución de problemas y ofreciendo apoyo personalizado [20].

En la literatura no hay un consenso entre la relación de asistente virtual y chatbot, para varios autores son dos términos que hacen referencia a la misma definición [21, 22], para Guijarro [23] existen tres diferencias fundamentales entre los asistentes virtuales y los chatbots: la forma de desarrollo de estas tecnologías es distinta, no llevan a cabo las mismas funciones y no se desarrollan en el mismo entorno. La principal diferencia es que los asistentes virtuales son creados por las grandes compañías tecnológicas con una ambición muy concreta: convertirse en un asistente personal, resolviendo todas las dudas que puedan surgir en cualquier momento.Otra de las diferencias es que si bien el objetivo de los asistentes virtuales es amplio, el de los chatbots es más específico. Los chatbots se crean con la finalidad de resolver una acción clara con un objetivo concreto. Suele estar enfocada a la resolución de preguntas frecuentes, aligerar el grueso de trabajo a los centros de contacto o la captación de posibles clientes. La otra diferencia se resume en que a pesar de que ambos han nacido como consecuencia del auge de la tecnología conversacional, chatbots y asistentes virtuales no se integran en los mismos entornos. Mientras que los asistentes virtuales nacen en sí mismos como una nueva aplicación en el móvil y se alojan en un entorno completamente novedoso que ya se conoce como Home Speakers o Home Devices, como Google Home, Amazon Echo o Home iPod de Apple, los chatbots se desarrollan en páginas webs, apps de mensajería, redes sociales o aplicaciones [23].

Para Granizo [6] los chatbots son un tipo de asistente virtual, generalmente entrenados para fines específicos, es decir, la información o tareas que realizan se encuentran delimitadas por la entidad que lo entrena, esto es lo que marca la diferencia con los asistentes virtuales de tipo personal como lo son Siri de Apple, Google Assistant de Google, Alexa de Amazon, o Cortana de Windows. Cerca de este criterio Rubio [24] señala que chatbot es una modalidad de implementar un asistente virtual. Expsósito [19] plantea que chatbot es una forma de asistente virtual que se distingue de los que son albergados en dispositivos electrónicos. Para esta investigación se tomará el criterio de que los chatbots son un tipo de asistente virtual para así distinguirlos de los asistentes virtuales de tipo personal.

En la literatura sobre chatbots, se presentan varias definiciones que destacan diversos aspectos de estas aplicaciones conversacionales; Rubio [25] describe a los chatbots como programas que pueden integrar inteligencia artificial, permitiéndoles simular una conversación humana natural. Esta definición resalta la capacidad de los chatbots de emular interacciones humanas de manera fluida y natural, gracias a la inteligencia artificial.

Galitsky [26] en su libro Developing Enterprise Chatbots: Learning Linguistic Structures, define al chatbot como un sistema informático que actúa como interfaz entre los usuarios y las aplicaciones de software, usando lenguaje natural como principal medio de comunicación. Este enfoque recalca el rol de los chatbots como intermediarios entre el usuario y otros sistemas, priorizando el lenguaje natural en la interacción.

Huang et al. [27] explican que un chatbot es un programa que interactúa con personas mediante texto o voz, ofreciendo respuestas rápidas a preguntas comunes y tareas repetitivas. Esta definición sugiere la eficiencia de los chatbots para gestionar solicitudes frecuentes y repetitivas, optimizando el tiempo y esfuerzo del usuario.

Según Abashev et al. [28] , los chatbots son programas computacionales que buscan mantener conversaciones de forma natural, simulando cómo interactuaría una persona. Para lograrlo, utilizan procesamiento de lenguaje natural .Esta definición se centra en la capacidad de los chatbots para simular interacciones humanas mediante herramientas de procesamiento del lenguaje natural, resaltando su función de crear una experiencia conversacional fluida.

Otra perspectiva Dibitonto et al. [29] describe a los chatbots como agentes de máquina que se comunican con humanos utilizando lenguaje natural. Esta definición ve a los chatbots como interfaces automatizadas, destacando su rol de agentes de comunicación entre usuarios y sistemas, lo que amplía su aplicabilidad en distintos contextos.

Skjuve et al. [30] describen a los chatbots como agentes conversacionales que permiten el acceso a datos y servicios , así como el intercambio de información mediante un diálogo en lenguaje natural. Este diálogo busca imitar una conversación humana, donde el chatbot analiza y responde basándose en palabras y frases individuales. Esta definición destaca la interacción bidireccional en tiempo real que permite a los usuarios acceder a servicios y datos.

El autor de este trabajo considera que todas estas definiciones son válidas y coinciden en aspectos clave: los chatbots son programas informáticos diseñados para interactuar con humanos mediante un lenguaje natural, ya sea hablado o escrito. Además, utilizan inteligencia artificial y procesamiento de lenguaje natural para emular una conversación, facilitando el acceso a información o servicios, optimizando la eficiencia en tareas repetitivas, y permitiendo una experiencia de usuario más accesible y personalizada.

### **Características de los chatbots**

Kuligowska et al. [31] propusieron un interesante método de evaluación del marco para identificar diferentes aspectos del funcionamiento de los chatbots, que incluye la apariencia visual, la base de conocimientos (información general y especializada porque un agente conversacional debe poder responder a un conjunto de preguntas de conocimiento general, no solo sobre el tema en el que está especializado, un aspecto cercano a los rasgos de personalidad del chatbot), habilidades conversacionales, habilidades lingüísticas y sensibilidad al contexto (un buen chatbot debe poder mantener un diálogo coherente, comprender el contexto, dar retroalimentación y reparar el diálogo si es necesario, para minimizar la frustración del usuario durante la interacción), rasgos de personalidad (un chatbot exitoso debe tener una personalidad única, incluidos aspectos de personalidad, biografía y emociones), respuestas de emergencia en situaciones inesperadas (el chatbot debe poder superar las faltas de ortografía del usuario u otros errores y responder diplomáticamente a provocaciones groseras o sexuales), posibilidad de calificar el chatbot (pedir directamente a los usuarios que califiquen la conversación es un valor agregado importante para estimar la apreciación del agente virtual) [31] .

Las características de los chatbots se pueden resumir de la siguiente forma:

* Inteligencia: Chaves y Gerosa [32] describen la inteligencia como la capacidad de un chatbot para participar en un diálogo con conciencia del tema discutido, mientras que Jain et al. [33] creen que la inteligencia de un chatbot también se puede deducir de su capacidad para hacer preguntas adecuadas de forma proactiva e involucrar al participante en un diálogo significativo y similar al humano.
* Interacción: Kiousis [34] define la interacción como el grado en el que una tecnología de comunicación puede crear un entorno mediado en el que los participantes pueden comunicarse , tanto de forma sincrónica como asincrónica, y participar en intercambios recíprocos de mensajes. En consonancia con esto, los chatbots permiten a las personas interactuar con los sistemas informáticos a través del lenguaje natural escrito y/o hablado con el objetivo de llevar la interacción de la forma más natural posible para que se asemeje a un diálogo cara a cara Diederich et al. [35].
* Contexto:el contexto en general es la totalidad de toda la información situacional implícita y explícita sobre personas, objetos, tiempo y ubicación dentro de una interacción que se puede utilizar para describir una situación [36] .El contexto muestra si el chatbot opera y en qué dominio [35].

### **Clasificación de los chatbots**

Los chatbots se pueden clasificar utilizando diferentes parámetros: el dominio de conocimiento, el servicio prestado, los objetivos, el método de procesamiento de entrada y generación de respuestas, la ayuda humana y el método de construcción [37] .

**Dominio de conocimiento** : la clasificación basada en el dominio de conocimiento considera el conocimiento al que puede acceder un chatbot o la cantidad de datos con los que está entrenado. Los chatbots de dominio abierto pueden hablar sobre temas generales y responder de manera apropiada, mientras que los chatbots de dominio cerrado se centran en un dominio de conocimiento en particular y pueden no responder a otras preguntas [37].

**Servicio proporcionado:** la clasificación basada en el servicio proporcionado considera la proximidad sentimental del chatbot con el usuario, la cantidad de interacción íntima que se lleva a cabo y también depende de la tarea que esté realizando el chatbot. Los chatbots interpersonales se encuentran en el dominio de la comunicación y brindan servicios como reservas de restaurantes, reservas de vuelos y bots de preguntas frecuentes. No son compañeros del usuario, pero obtienen información y se la transmiten al usuario. Pueden tener personalidad, pueden ser amigables y probablemente recordarán información sobre el usuario, pero no están obligados ni se espera que lo hagan. Los chatbots intrapersonales existen dentro del dominio personal del usuario, como aplicaciones de chat como Messenger, Slack y WhatsApp. Son compañeros del usuario y lo entienden como lo hace un humano. Los chatbots entre agentes se vuelven omnipresentes, mientras que todos los chatbots requerirán algunas posibilidades de comunicación entre chatbots. Ya ha surgido la necesidad de protocolos para la comunicación entre chatbots. La integración de Alexa-Cortana es un ejemplo de comunicación entre agentes [37].

**Objetivos:** la clasificación basada en los objetivos considera el objetivo principal que los chatbots pretenden alcanzar. Los chatbots informativos están diseñados para proporcionar al usuario información que se almacena de antemano o está disponible en una fuente fija, como los chatbots de preguntas frecuentes. Los chatbots basados en chat/conversacionales hablan con el usuario, como otro ser humano, y su objetivo es responder correctamente a la oración que se le ha dado. Los chatbots basados en tareas realizan una tarea específica, como reservar un vuelo o ayudar a alguien. Estos chatbots son inteligentes en el contexto de solicitar información y comprender la entrada del usuario. Los bots de reserva de restaurantes y los chatbots de preguntas frecuentes son ejemplos de chatbots basados en tareas [38].

### **Arquiteturas de chatbots**

**Sistemas basados en reglas:** los chatbots basados en reglas son el tipo de arquitectura con el que se han construido la mayoría de los primeros chatbots, como numerosos chatbots en línea. Eligen la respuesta del sistema basándose en un conjunto fijo de reglas predefinidas, basadas en el reconocimiento de la forma léxica del texto de entrada sin crear nuevas respuestas de texto. El conocimiento utilizado en el chatbot está codificado a mano por humanos y se organiza y presenta con patrones conversacionales [39]. Una base de datos de reglas más completa permite al chatbot responder a más tipos de entrada del usuario. Sin embargo, este tipo de modelo no es robusto a los errores ortográficos y gramaticales en la entrada del usuario. La mayoría de las investigaciones existentes sobre chatbots basados en reglas estudian la selección de respuestas para conversaciones de un solo turno, que solo consideran el último mensaje de entrada. En chatbots más parecidos a los humanos, la selección de respuestas de múltiples turnos tiene en cuenta partes anteriores de la conversación para seleccionar una respuesta relevante para todo el contexto de la conversación [40].

**Sistemas basados en recuperación:** los chatbots basados en recuperación utilizan respuestas predefinidas almacenadas en una base de datos para generar respuestas a las entradas de los usuarios. Estos sistemas funcionan al hacer coincidir las entradas del usuario con la respuesta más cercana en la base de datos, utilizando métricas de similitud como la similitud coseno o coincidencias de palabras clave. Los sistemas basados en recuperación pueden manejar un rango más amplio de entradas en comparación con los sistemas basados en reglas y pueden proporcionar respuestas más relevantes según el contexto. Sin embargo, están limitados por la calidad y cobertura de la base de datos de respuestas y pueden tener dificultades para generar respuestas nuevas [41, 42].

**Modelos generativos:** las arquitecturas de chatbots generativos utilizan modelos de aprendizaje automático y técnicas de aprendizaje profundo , para generar respuestas a las entradas de los usuarios. Estos modelos se entrenan en grandes conjuntos de datos de conversaciones y aprenden a generar respuestas basadas en el contexto de la conversación. Los modelos generativos pueden producir respuestas más diversas y relevantes en comparación con los sistemas basados en reglas o en recuperación. Sin embargo, requieren grandes cantidades de datos de entrenamiento y recursos computacionales, y pueden tener dificultades para mantener la coherencia y relevancia en conversaciones largas  [42 , 43].

## **Procesamiento del Lenguaje Natural**

El Procesamiento del Lenguaje Natural (PLN) [43] es una rama de la inteligencia artificial que ayuda a las máquinas a entender, interpretar y manipular el lenguaje natural, así como determinar la acción adecuada y responder en el mismo idioma del usuario. Según Deng y Liu [44] en su libro *Deep Learning in Natural Language Processing* indican que “desde una perspectiva de ingeniería, el PLN se preocupa por cómo desarrollar aplicaciones prácticas novedosas para facilitar las interacciones entre las computadoras y los lenguajes humanos”. Los motores PLN se encargan de recibir las intenciones del usuario, procesarlas, interpretarlas y eventualmente producir una respuesta en lenguaje natural al humano.

Para que esto sea posible, se lleva a cabo, mediante algoritmos, la conversión de los datos no estructurados, es decir el lenguaje natural, en datos estructurados que pueden ser usados por las máquinas. Algunas de las tareas básicas del PLN son la clasificación de textos por temática, el etiquetado de partes de un discurso, la detección del lenguaje, la identificación de relaciones semánticas, traducciones automáticas, y chatbots [19].

La arquitectura de un sistema NLP contienen varios niveles de estudio como lo son [45] :

1. Nivel Fonológico: trata como las palabras se relaciona con los sonidos que representan.
2. Nivel Morfológico: examina la parte interna de la palabra.
3. Nivel Sintáctico: estudia la estructura de la oración mediante patrones.
4. Nivel Semántico: interpreta el significado de cada oración.
5. Nivel Pragmático: analizar y agrupa todas las oraciones para comprender el contexto del diálogo.

Dentro del NLP se puede encontrar dos modelos [46] :

1. Modelo Lógico: estudia la gramática del lenguaje, es decir su estructura abarcando las reglas y diccionarios que permite entender las expresiones del lenguaje nativo.
2. Modelo Probabilístico: examinan los corpus (conjunto de textos) que contienen algoritmos que son capaces de predecir respuestas que son adquiridas por el aprendizaje automático que según plantea Samuel [47] es el campo de estudio que da a las compudadoras la capacidad de aprender sin ser programadas de manera explícita.

El PLN abarca todas las técnicas necesarias para operar con el lenguaje natural, dentro de las cuales se pueden distinguir dos grupos o subramas. En primer lugar, se encuentra la rama de *Natural Language Understanding (NLU)*: en español significa Comprensión de Lenguaje Natural, la cual se encarga de averiguar el significado e intención que encierra un mensaje. Funciona a través de algoritmos de aprendizaje profundo, análisis estadístico del orden y frecuencia de las palabras y una gran cantidad de datos de capacitación. El uso de NLU en los chatbots permite mejorar la experiencia de usuario, ya que la conversación es más fluida, sin tener la necesidad de usar respuestas preestablecidas como sucede en el caso del resto de tecnologías PLN. Y por otro lado, se encuentra la *Natural Language Generation (NLG)* que en español significa Generación del Lenguaje Natural que se basa en recrear de manera automática el lenguaje natural a partir de los datos estructurados con los que trabaja el computador, con el fin de que el ser humano pueda entenderlos. Su uso es fundamental en el ámbito de los chatbots para permitir la comunicación tanto por texto como por voz [19].

## **Modelo de Lenguaje Grande**

Un Modelo de Lenguaje Grande o *Large Language Model (LLM)* es un modelo basado en redes neuronales las cuales son algoritmos implementados en forma de programa informático o modelo electrónico, influenciados por el funcionamiento del cerebro humano y la principal propiedad es la capacidad de aprender del entorno en el que opera y mejorar su funcionamiento [48].

Para el caso de los LLMs son basados en redes neuronales de tipo *Transformer*, entrenado con un corpus extenso de datos. Su objetivo principal es modelar patrones del lenguaje para abordar tareas específicas en el PLN, lo cual implica la comprensión, el procesamiento y la manipulación del lenguaje humano, así como la generación de lenguaje natural.

Anteriormente, las redes neuronales recurrentes eran las encargadas de estas tareas, ya que trataban el texto como una secuencia de caracteres [49]. Sin embargo, estas redes estaban limitadas a recordar solo unas pocas palabras previas en la secuencia, lo que hacía que perdieran el contexto. En este sentido, surgieron las redes *Transformer*, diseñadas específicamente para abordar esta limitación.

Las redes *Transformer* son una arquitectura de red neuronal desarrollada por Google en 2017. A diferencia de las redes recurrentes, estas poseen una memoria de mayor alcance gracias a los mecanismos de self-attention y multi-head attention, los cuales procesan la información de forma paralela. Esta arquitectura fue presentada en el artículo *“Attention is All You Need”* de Vaswani et al. [49]. Muchos modelos de lenguaje actuales usan esta arquitectura para abordar tareas de procesamiento y generación de lenguaje natural [50].

Un modelo de lenguaje es un modelo computacional capaz de entender y generar lenguaje natural; un LLM es un modelo de lenguaje entrenado con una cantidad de datos considerablemente mayor. Un ejemplo destacado es ChatGPT, un sistema de chat basado en el modelo de lenguaje GPT-3.5 de OpenAI [50].

A pesar de haber sido entrenados con grandes cantidades de datos, los LLMs presentan algunas limitaciones. Entre ellas se encuentran la falta de información en tiempo real y el entrenamiento en fuentes genéricas, lo que puede resultar en respuestas poco técnicas o desactualizadas. Estas limitaciones pueden ser problemáticas en aplicaciones empresariales, como en el caso de Wattwin. Los LLMs especializados en dominios específicos deben recibir datos propietarios y específicos del negocio para proporcionar respuestas precisas y contextualizadas sobre procesos empresariales particulares, incluso si estos datos no fueron utilizados en su entrenamiento. Para abordar estas limitaciones, surge la arquitectura de Generación Aumentada por Recuperación [51].

## **Generación Aumentada por Recuperación**

La técnica de Generación Aumentada por Recuperación o *Retrieval Augmented Generation (RAG)* se basa en mejorar el conocimiento de un LLM proporcionándole contexto adicional, sin necesidad de reentrenar el modelo ni de crear uno especializado para un área concreta. RAG combina dos conceptos clave: primero, utiliza representaciones vectoriales de documentos para indexarlos y extraer la información más relevante para cada consulta; segundo, emplea esta información como contexto para un LLM entrenado con datos genéricos. La integración de documentación adicional en los LLM busca mejorar la precisión y confiabilidad de las respuestas, mitigando posibles "alucinaciones" [51].

La arquitectura RAG optimiza los LLM mediante la incorporación de documentos que aportan conocimiento adicional. Los documentos de diversas fuentes se fragmentan para facilitar su procesamiento; luego, un modelo de embedding crea representaciones vectoriales que se almacenan en una base de datos vectorial. Al formular una pregunta, se genera su representación vectorial con el mismo modelo de embedding, y algoritmos de distancia como la similitud de coseno asignan una puntuación a cada fragmento para determinar su relevancia en relación con la pregunta. Los fragmentos más relevantes, es decir, aquellos con mayor similitud semántica, se incorporan al prompt, un conjunto de instrucciones para guiar la generación de texto del LLM, incluyendo tanto la pregunta inicial del usuario como los textos relevantes [50].

La arquitectura RAG enfrenta dos problemas principales: la gestión de información en bases de datos y la ejecución de búsquedas semánticas en cada interacción con el LLM. En primer lugar, aplicar la técnica de chunking a bases de datos presenta limitaciones como [50]:

1. Número de fragmentos recuperados: a diferencia de los documentos en PDF, donde los fragmentos pueden tratarse de forma independiente, en bases de datos se deben recuperar todas las filas relevantes para una consulta específica, lo que dificulta saber cuántos fragmentos se necesitan.
2. Actualizaciones en tiempo real: los datos de una base de datos cambian frecuentemente. Para ofrecer respuestas precisas, es necesario actualizar los fragmentos y sus representaciones vectoriales cada vez que se modifica, inserta o elimina un registro, reflejando los cambios de inmediato.

Además, realizar una búsqueda semántica en cada interacción del chat puede ser innecesario en algunos casos, como cuando el usuario solo saluda. En tales situaciones, una respuesta directa sin búsqueda adicional ahorra tiempo y costos de procesamiento, evitando la recuperación innecesaria de documentos y la adición de información extra al prompt. La recuperación constante de información también puede introducir datos irrelevantes, lo cual afecta la precisión de las respuestas [50].

Enfrentando estos retos aparece el concepto de agente, un enfoque que permite a un LLM determinar los pasos a seguir para responder a una tarea específica [52]. La idea central detrás de un agente es emplear un modelo de lenguaje para razonar sobre la secuencia de pasos necesarios para responder preguntas o realizar acciones. A diferencia de una cadena de acciones lineal, en la cual se siguen todos los pasos incluso si no son necesarios, un agente decide qué tareas realizar y en qué orden [50].

## **Agentes Inteligentes**

Se han propuesto diversas definiciones de agentes inteligentes, cada una de ellas con el objetivo de capturar el significado y la aplicación de dicho término . Estas definiciones varían en su complejidad y alcance, reflejando la diversidad de enfoques y contextos en los que se utilizan los agentes [53] . En un enfoque basado en agentes, estos son tratados como las entidades principales, similares a cómo los objetos son fundamentales en los enfoques orientados a objetos. En un entorno donde se utilizan agentes para programar, los programas se desarrollan al definir los comportamientos, objetivos e interacciones de dichos agentes, reflejando así su funcionalidad en el contexto computacional [54].

Según Russell y Norvig [1], un agente es cualquier entidad que pueda percibir su entorno a través de sensores y actuar sobre ese entorno a través de actuadores. La definición de estos autores proporciona un marco general que abarca tanto sistemas físicos como de software. Destaca la percepción a través de sensores y la acción a través de actuadores como los elementos clave de la interacción del agente con el entorno. Esta visión distingue claramente la capacidad del agente para adaptarse, percibir y actuar en su entorno, independientemente de su naturaleza física o digital.  Un agente inteligente debería seleccionar acciones que se espera que maximicen los criterios que determinan el éxito en su comportamiento. Esto implica que el agente, al enfrentar una secuencia de percepciones, debería evaluar y elegir aquella acción que se espera que conduzca al mejor resultado posible según una medida de rendimiento predefinida. La selección de acciones a lo largo del tiempo se basa en la optimización de estas medidas de rendimiento [1] .

En la perspectiva de Maes [55], los agentes autónomos son sistemas computacionales que residen en un entorno dinámico y complejo, donde son capaces de percibir y actuar de manera autónoma para cumplir con un conjunto de objetivos o tareas específicas para las que han sido diseñados. En esta definición, destaca el elemento crucial de la autonomía de los agentes al afirmar que deben actuar por sí mismos para lograr sus objetivos. Además, se enfatiza que los agentes autónomos operan en entornos que cambian mientras el agente toma decisiones.

Por otra parte la Fundación para Agentes Físicos Inteligentes (FIPA) [56] , plantea que un agente es un proceso computacional que implementa la capacidad autónoma de comunicación en una aplicación, es decir que un agente puede interactuar con otros agentes o entidades dentro de un sistema de manera independiente, sin una supervisión constante o intervención externa.

### **Características de los agentes inteligentes**

Según Wooldridge y Jennings [57] un agente es un sistema informático basado en hardware y/o software que muestra las características:

* Autonomía: los agentes operan sin la intervención directa de humanos u otros, y tienen algún tipo de control sobre sus acciones y estado interno.
* Habilidad social: los agentes tienen la capacidad de interactuar con otros agentes y posiblemente con humanos mediante un lenguaje de comunicación de agentes.
* Reactividad: los agentes son capaces de detectar su entorno y reaccionar de manera rápida y adecuada ante cualquier modificación que ocurra en él.
* Proactividad: los agentes muestran un comportamiento orientado a metas al tomar la iniciativa propia para actuar, en lugar de simplemente reaccionar a cambios en su entorno.

## **Chatbots en bibliotecas universitarias**

Muchas áreas, incluida la educación, han cambiado mucho debido a la rapidez con que ha avanzado la tecnología en los últimos diez años. Uno de estos cambios es que los servicios bibliotecarios están utilizando cada vez más la inteligencia artificial  y la tecnología. Las bibliotecas universitarias están adaptándose para satisfacer las demandas de la era moderna. En el pasado, eran conocidas por ser lugares donde las personas podían encontrar información y datos. Este cambio se refleja en la introducción de nuevas tecnologías que buscan mejorar la experiencia del usuario, optimizar los servicios y hacer las operaciones más eficientes. El uso de chatbots automatizados es un ejemplo de este tipo de avance tecnológico. Estos sistemas impulsados por IA están diseñados para brindar ayuda en tiempo real, facilitar las interacciones con los usuarios y ofrecer apoyo personalizado, lo que podría transformar la manera en que las bibliotecas universitarias atienden a sus usuarios [58].

Varias universidades han utilizado con éxito chatbots para mejorar sus servicios bibliotecarios, demostrando el potencial y uso adecuado de esta tecnología. La Universidad de Oklahoma es un buen ejemplo, ya que integraron un chatbot llamado "Bizzy" para responder preguntas bibliotecarias. Bizzy funciona junto con el sistema de gestión bibliotecaria de la universidad y proporciona respuestas en tiempo real a preguntas frecuentes, ayuda a los usuarios a navegar por los recursos digitales y sugiere libros de interés. Los usuarios han elogiado la rapidez y precisión de Bizzy, lo cual ahorra tiempo y esfuerzo al personal de la biblioteca al manejar consultas rutinarias [59].

Otro notable ejemplo es la Universidad de California, Irvine (UCI), que desarrolló un chatbot llamado "ANTswers" para mejorar la interacción con los usuarios y ofrecer asistencia adicional [58]. ANTswers utiliza PLN para comprender y responder a una amplia variedad de consultas de los usuarios. Los estudiantes pueden usarlo para encontrar recursos académicos en la extensa base de datos de la biblioteca, obtener respuestas a sus preguntas de investigación y aprender sobre los servicios y políticas de la biblioteca. Desde su implementación, ANTswers ha facilitado el acceso a los recursos bibliotecarios, especialmente para los usuarios que no se encuentran en el mismo espacio físico, y ha sido una pieza clave para ofrecer soporte las 24 horas [58].

Con su robot "Inquire Us," disponible tanto en el sitio web de la biblioteca como en la aplicación móvil, la Universidad Estatal de Carolina del Norte (NCSU) ofrece otro ejemplo destacado. Inquire Us utiliza IA para brindar respuestas rápidas y precisas a todo tipo de preguntas, como la localización de libros y materiales digitales o el entendimiento de normas y servicios bibliotecarios. El chatbot actualiza constantemente su información, de manera que los usuarios siempre reciben ayuda actualizada y útil. Inquire Us en NCSU ha incrementado la satisfacción de los usuarios y también ha permitido que el personal docente se enfoque en tareas más complejas y relevantes [58].

Tanto la Universidad Estatal de San José como las Bibliotecas de la Universidad Estatal de Pensilvania comenzaron a utilizar chatbots. En septiembre de 2020, la Universidad Estatal de San José lanzó Kingbot, creado con Kommunicate, un software propietario que utilizaba Dialogflow de Google. Kingbot estaba activo de 7 PM a 9 AM en días de semana y de 5 PM a 1 PM los fines de [60]. En febrero de 2023, las Bibliotecas de la Universidad Estatal de Pensilvania lanzaron su chatbot de biblioteca, que utilizaba Springshare y estaba activo cuando un operador en vivo no estaba en línea [61].

La Universidad de Pretoria en Sudáfrica es otro caso exitoso en el uso de chatbots. Su robot, "Libby," fue diseñado para asistir tanto a estudiantes como a profesores, respondiendo preguntas comunes sobre la biblioteca y facilitando el acceso a materiales académicos. Libby ha sido especialmente útil para los estudiantes que estudian a distancia, ya que está disponible las 24 horas del día, los 7 días de la semana, para responder preguntas sobre la biblioteca [58].

El proyecto titulado “Bullbot: Asistente virtual para la gestión de servicios de la biblioteca de la ULL” [19] desarrollado en España como trabajo de fin de grado de la Universidad de la Laguna. Su objetivo fue elaborar un chatbot que sea capaz de gestionar los servicios de la biblioteca. El asistente actúa como una interfaz que facilita a los usuarios el acceso a la información de la web oficial de manera cómoda y natural. El proyecto en cuestión se desarrolló haciendo uso intensivo de la tecnología de inteligencia artificial de IBM. Este trabajo tiene soluciones que satisfacen la mayoría de las necesidades de los usuarios, pero usa tecnología IBM que es de pago.

Otro ejemplo es “Propuesta de Chatbots para mejorar los servicios digitales de la Biblioteca Octavio Arizmendi Posada de la Universidad de La Sabana” [18] en Colombia, este proyecto tuvo como propósito diseñar un asistente virtual de tipo chatbots para contribuir a la optimización de los servicios digitales de dicha biblioteca. El chatbots fue implementado en la interfaz de la página web de la biblioteca cumpliendo con los servicios de capacitaciones, prestamos, y reservas a cabalidad.

## **Tecnologías para el desarrollo de chatbots**

Muchas de las grandes empresas de la industria tecnológica ofrecen plataformas de procesamiento de lenguaje natural como es el caso de Google con Dialogflow, Amazon con Lex, IBM con Watson, entre otras, cada una de las propuestas de estas plataformas tiene el mismo concepto, pero diferentes enfoques o alcances [62], por eso se procede a investigar a fondo aspectos clave de las  plataformas principales, para evaluar las opciones y tomar la mejor decisión de cual herramienta usar para el desarrollo de este trabajo.

### **Dialogflow**

Dialogflow es una herramienta fácil e intuitiva, capaz de entender y procesar el lenguaje natural, permite implementar intenciones anidadas lo cual facilita el hilo conversacional y dificulta la pérdida de contexto dentro de la conversación, ya que permite hacer una pregunta o afirmación dentro de otra, su flexibilidad en el diseño de conversaciones demuestra su versatilidad, además Google introdujo Knowledge Connectors, un sistema capaz de comprender y extraer de forma automática las preguntas y las respuestas del contenido que previamente fue implementado por el desarrollador. Además pretende eliminar el obstáculo de las fallas ortográficas mediante la inclusión del plugin Automatic Spelling Correction. Para su funcionamiento define unas palabras clave llamadas entidades, las cuales ayudan a identificar la necesidad de los usuarios, asociándolas a una intención y a llevar el hilo conversacional a través de un contexto. Cuenta con una herramienta llamada Fulfillment, la cual es un código que se implementa como un webhook y permite al agente de Dialogflow generar respuestas dinámicas o desencadenar acciones en su back-end o base de datos. Ya que es una plataforma de Google, se integra fácilmente con Google Assistant para que la API sea usada desde altavoces inteligentes, dispositivos móviles y casi cualquier plataforma de mensajería. Se puede configurar y personalizar para ser usada en cualquier página web, adaptándose al diseño de la interfaz. Estas son algunas de las plataformas con las cuales se puede integrar Dialogflow con tan solo un clic [63].

### **IBM Watson**

Por otro lado, IBM Watson tiene un editor de diálogo visual, lo que significa que personas sin conocimiento de programación podrán desarrollar un chatbot sin ningún problema, además permite crear interfaces de conversación en cualquier aplicación, dispositivo o canal. Watson funciona con conocimientos de diálogo, datos de entrenamiento y el uso de lógica que permiten al asistente interpretar la petición del usuario. Los conocimientos contienen tres tipos de artefactos las intenciones que representan la finalidad de la entrada del usuario a partir del reconocimiento de una palabra clave llamada entidad y el diálogo que es el hilo de la conversación. Los bot creados son gestionados completamente por IBM Cloud por lo que el usuario no tiene que preocuparse por configurar o dar soporte a la infraestructura. Uno de los puntos más fuertes de IBM Watson es el servicio de reconocimiento de imágenes, principalmente para un asistente de carrito de compras o tienda virtual, ya que permite entender con mayor facilidad las intenciones del usuario y es muy fácil de implementar, solo se requiere una carpeta de imágenes diferentes del articulo a reconocer y ya está. En el momento de la creación de un bot, Watson permite elegir un ejemplo prediseñado para trabajar sobre él, cada uno describe su funcionamiento básico en la presentación, lo único que tendrá que hacer el creador es desarrollar el diálogo con posibles preguntas hechas por el usuario y las respuestas que tendrá que dar la plataforma, ya que la herramienta se encarga de procesar el texto de entrada, buscar palabras clave que ayuden a identificar la intención para ser ejecutada [64].

### **Amazon Lex**

Amazon Lex, funciona con el mismo motor conversacional que usa Amazon Alexa a través de unas variables de sesión que se utilizan para almacenar el estado de la conversación, incluidas las entradas de usuarios, Lex no maneja contextos para conservar el hilo de la conversación, pero funciona con el llenado de slots o parámetros como operación fundamental. Estos parámetros pueden ser nombres, ciudades, tiempo entre otros, que pueden ser almacenadas y luego usadas para dar una respuesta. Esta herramienta no tiene costos por anticipado ni cuotas mínimas. Solo se le cobrará por las solicitudes de texto o voz que se realicen. Amazon Lex permite publicar el bot desarrollado en plataformas de mensajería como Messenger, Slack y Twilio SMS. Amazon Lex también se integra con AWS Mobile Hub, por lo que puede lograr que una aplicación móvil acceda a un bot de conversación de Lex de manera rápida. Al ser una plataforma de la empresa Amazon, Lex dispone de integración incorporada con AWS Lambda, AWS MobileHub y Amazon CloudWatch. Además, puede integrarlo de manera sencilla con muchos otros servicios de la plataforma de AWS, como Amazon Cognito y Amazon DynamoDB. Esta herramienta puede exportar bots, intenciones y tipos de slot para compartirlos o modificarlos, los recursos deben exportarse en un formato compatible con el servicio al que exporta, ya sea Amazon Lex o Alexa Skills Kit. Lex al ser una plataforma de Amazon, requiere credenciales válidas para autenticar las solicitudes, si no tiene permisos, no podrá crear los recursos de Amazon Lex ni obtener acceso a ellos por su robusto sistema de control de acceso [65].

### **Rasa**

En contraste Rasa, es un marco de aprendizaje automático de código abierto basado en python para crear asistentes contextuales de AI y chatbots. Incluye una interfaz de usuario y una API REST. Funciona con un conjunto de herramientas de aprendizaje automático. Tiene dos módulos principales: NLU para entender los mensajes de usuario. Núcleo para mantener conversaciones y tomar decisiones.Para construir un asistente con Rasa, es necesario instalar Rasa NLU y Rasa Core. También es necesario instalar un modelo de idioma spaCy. Luego se debe crear un proyecto que contiene la estructura de archivos habitual, algunos datos de entrenamiento y los archivos principales que se pueden utilizar como base para crear un asistente personalizado [66].

### **Langchain**

Langchain es una herramienta de código abierto basada en python clave para construir aplicaciones que integran modelos de lenguaje  y técnicas avanzadas de procesamiento de información, facilitando la orquestación de interacciones entre diferentes componentes. Su diseño permite la integración de LLMs con la recuperación de información y agentes autónomos, lo que es esencial para mejorar la precisión y contextualidad de las respuestas generadas. Langchain facilita el acceso a bases de datos vectoriales que almacenan embeddings de documentos, permitiendo realizar búsquedas semánticas. Además, permite la creación de agentes que pueden ejecutar tareas específicas e interactuar dinámicamente con los usuarios, consultando bases de datos y generando respuestas adaptadas a consultas complejas. Este mecanismo optimiza la experiencia del usuario y el manejo de datos, posicionando a Langchain como una herramienta fundamental en el desarrollo de asistentes virtuales personalizados basados en inteligencia artificial [52].

## **Selección de la tecnología pra el desarrollo del chatbot**

### **Langchain**

Para el desarrollo del chatbot de la biblioteca universitaria, se ha seleccionado Langchain, una tecnología de código abierto que permite integrar modelos LLMs en un entorno flexible y altamente personalizable. Langchain destaca por su capacidad para orquestar interacciones avanzadas entre los componentes de procesamiento de lenguaje natural y recuperación de información, lo que es ideal para un entorno académico. Además, admite la arquitectura de RAG, una metodología que permite enriquecer las respuestas generadas mediante la recuperación de documentos específicos de una base de datos vectorial, garantizando así que las respuestas sean contextualizadas y relevantes para las consultas de los usuarios.

En comparación, herramientas propietarias como Dialogflow, IBM Watson y Amazon Lex ofrecen menos flexibilidad debido a sus integraciones y dependencias dentro de sus respectivos ecosistemas. Por ejemplo, Dialogflow se integra de forma óptima con Google Cloud y Google Assistant, mientras que Amazon Lex funciona especialmente bien dentro del entorno AWS. Estas plataformas requieren suscripciones y suelen tener limitaciones en cuanto a la personalización, lo cual puede ser restrictivo para el desarrollo de un chatbot especializado como el de la biblioteca universitaria. IBM Watson, aunque proporciona un editor visual que facilita la creación de interfaces conversacionales, está vinculado a IBM Cloud y puede no ser tan adecuado para un sistema académico autónomo debido a los costos y al control limitado sobre el procesamiento y almacenamiento de datos.

Por otra parte, Rasa, también de código abierto, es una opción robusta que permite construir chatbots contextuales y personalizados. Sin embargo, Langchain proporciona una ventaja particular en el manejo de modelos LLMs y la arquitectura RAG, que son esenciales para un chatbot de biblioteca que necesite consultar grandes cantidades de texto y generar respuestas precisas. Langchain facilita además el uso de modelos LLM como los de Ollama, una plataforma que permite alojar y gestionar modelos de lenguaje [67] ,optimizando aún más la capacidad del chatbot para manejar consultas complejas y específicas de una manera dinámica y sin restricciones de costos adicionales asociados a tecnologías propietarias. Esta flexibilidad y adaptabilidad convierten a Langchain en la mejor opción para el desarrollo del chatbot de la biblioteca, asegurando que pueda evolucionar y adaptarse a las necesidades cambiantes del entorno académico. Este enfoque resulta ideal para un chatbot bibliotecario, ya que mejora la precisión y relevancia de las respuestas al combinar búsquedas en bases de datos vectoriales (almacenando embeddings de los documentos) con el poder de los LLMs. Esto permite al chatbot ofrecer recomendaciones personalizadas y respuestas detalladas sobre el catálogo bibliográfico y otros recursos académicos de la universidad, adaptándose a las necesidades de los estudiantes y el personal académico de manera eficaz.

### **Modelo de generación de embeddings de texto**

Para el proyecto, se selecciona el modelo Nomic Embed [68] como solución para la generación de embeddings de texto debido a varias razones específicas que se alinean con los requisitos del sistema. Este modelo, diseñado para contextos largos con una capacidad de hasta 8192 tokens, es particularmente adecuado para consultas complejas y recuperación de información en grandes bases de datos de textos, como los catálogos bibliográficos​.

Además, Nomic Embed supera a modelos populares como text-embedding-ada-002 en benchmarks clave como el Massive Text Embedding Benchmark (MTEB) (Figura 1 [68]) ofreciendo una mejor precisión en tareas de búsqueda semántica y recuperación aumentada por generación​. Su diseño de código abierto, junto con la publicación de sus datos de entrenamiento y pesos bajo una licencia Apache-2.0, garantiza una total transparencia y audibilidad, algo fundamental para un entorno académico donde la integridad y trazabilidad de los datos son críticas​.

Por otra parte, el modelo fue entrenado utilizando aprendizaje contrastivo en un corpus curado, lo que le permite manejar eficientemente datos no estructurados como preguntas y respuestas. Esto resulta vital en un chatbot que debe interpretar tanto consultas simples como consultas derivadas del historial de interacciones del usuario​.

Finalmente, la disponibilidad del modelo para uso local o a través de una API facilita su integración en la infraestructura de la biblioteca. Estas características hacen de Nomic Embed una elección sólida para optimizar las funcionalidades de recuperación y recomendación del chatbot para la biblioteca universitaria.

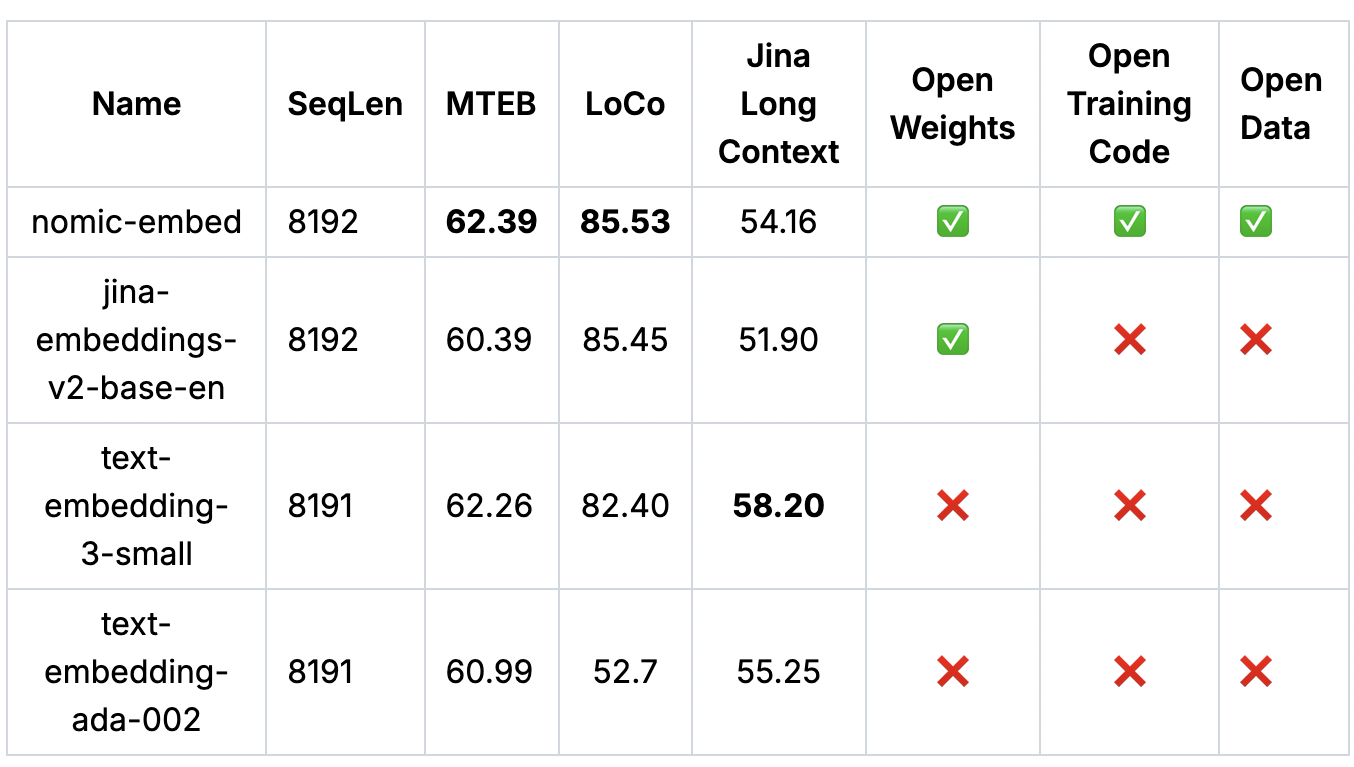


Figura 1: Massive Text Embedding Benchmark

### **LLM**

Se seleccionó Llama 3.1 (8B) [69] como modelo de lenguaje para el chatbot de la biblioteca debido a su equilibrio entre capacidad avanzada y eficiencia computacional. Este modelo, desarrollado por Meta, ofrece un rendimiento competitivo en tareas de generación de texto y comprensión contextual, mientras mantiene requerimientos moderados de recursos, lo que lo hace ideal para implementaciones prácticas en tiempo real.

Llama 3.1 (8B) ofrece un sobresaliente desempeño en tareas generales, razonamiento, matemáticas y uso de herramientas, todo esto en un modelo de tamaño manejable que optimiza recursos sin sacrificar calidad (Figura 2 [69]) . En el benchmark MMLU (0-shot, CoT), obtiene un puntaje de 73.0, superando modelos como Gemma 2 (9B IT) y Mistral (7B Instruct), mostrando su capacidad para responder preguntas de conocimiento general de manera efectiva.

En tareas de razonamiento, destaca en el desafío ARC Challenge (83.4), posicionándose cerca de modelos significativamente más grandes y robustos como GPT-3.5 Turbo (83.7), mientras mantiene costos de inferencia más bajos. Además, en GSM8K (matemáticas, 8-shot, CoT), logra 84.5, superando claramente a Mistral 7B Instruct y otros modelos similares, demostrando su habilidad para manejar problemas complejos.

Un aspecto diferenciador es su desempeño en el uso de herramientas, como lo refleja su puntaje de 76.1 en BFCL, lo cual es relevante para un chatbot que necesita interactuar con sistemas externos, como bases de datos vectoriales o APIs. Esto habilita a los agentes para realizar tareas específicas como búsquedas avanzadas y análisis de datos. Su carácter abierto y su soporte para frameworks como LangChain garantizan una integración fluida, personalización y alineación con los valores de transparencia y reproducibilidad necesarios en un entorno académico.

Finalmente, su excelente desempeño en tareas multilingües (Multilingual MGSM: 68.9) y manejo de contexto largo (hasta 32,000 tokens) lo convierten en una opción ideal para interactuar con usuarios diversos y manejar interacciones complejas, características fundamentales para un chatbot en una biblioteca.

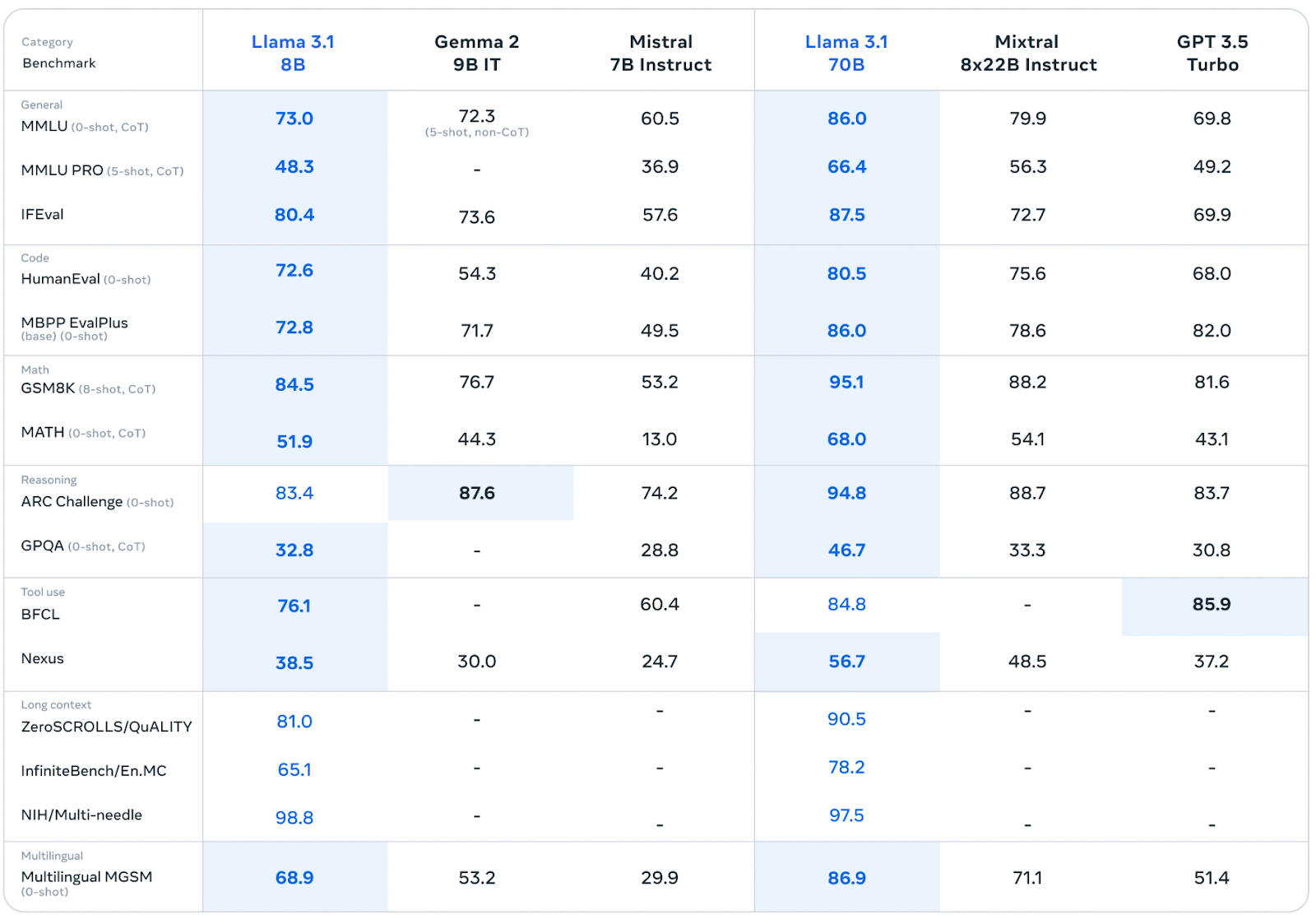


Figura 2: Comparación de Llama 3.1 8B con otros LLMs

### **Ollama**

Ollama [67] es una plataforma diseñada para alojar modelos de lenguaje localmente, brindando una solución robusta para proyectos que requieren control total sobre los datos y privacidad. Se seleccionó para este proyecto debido a su capacidad para integrar y gestionar eficientemente modelos avanzados como Llama 3.1 y Nomic Embed Text. Al alojar estos modelos de manera local, Ollama elimina la necesidad de depender de servidores externos, lo que reduce riesgos de seguridad y garantiza una baja latencia en las respuestas. En el caso de Llama 3.1, Ollama permite aprovechar su alta precisión para ofrecer respuestas contextuales y relevantes en el chatbot de la biblioteca, mientras que con Nomic Embed Text facilita la creación de embeddings de alta calidad para el análisis y recomendación de libros. Esta elección combina seguridad, rendimiento y funcionalidad, adaptándose perfectamente a las necesidades del proyecto.

### **Chroma DB**

Chroma DB [70] es una base de datos vectorial diseñada para almacenar, gestionar y realizar consultas sobre representaciones vectoriales de datos, como las generadas por modelos de aprendizaje automático. Está especialmente optimizada para tareas de búsqueda de similitud, lo que la convierte en una opción ideal para aplicaciones que necesitan trabajar con grandes volúmenes de datos no estructurados, como texto, imágenes o audio. Esta base de datos facilita el almacenamiento eficiente y la recuperación de información basada en vectores, lo que es crucial para sistemas como el que estás desarrollando, que maneja documentos y colecciones de libros.

En comparación con otras bases de datos vectoriales, Chroma DB presenta ventajas claras que justifican su selección para este proyecto. Por ejemplo, mientras que bases de datos como Pinecone [71] o Weaviate [72] también ofrecen funcionalidades avanzadas para el almacenamiento y consulta de vectores, Chroma DB destaca por ser de código abierto, lo que elimina costos asociados a licencias o suscripciones y permite una personalización completa según las necesidades específicas del sistema. Esto es particularmente relevante cuando se busca flexibilidad y control total sobre la infraestructura.

Además, Chroma DB proporciona una integración sencilla y eficiente con herramientas populares como LangChain, lo que facilita el desarrollo de aplicaciones basadas en inteligencia artificial, como asistentes virtuales o motores de recomendación. Por otro lado, bases de datos como Milvus [73] ofrecen escalabilidad robusta para grandes volúmenes de datos, pero pueden requerir una mayor inversión en términos de configuración y mantenimiento. En contraste, Chroma DB es más accesible para desarrolladores que buscan un equilibrio entre funcionalidad avanzada y facilidad de uso.

Otra consideración importante es el enfoque de rendimiento. Si bien opciones como FAISS [74] son altamente eficientes para búsquedas en memoria y funcionan bien en entornos controlados, estas suelen carecer de características más completas para la gestión y persistencia de datos a largo plazo. Chroma DB, en cambio, combina un rendimiento sólido con capacidades de almacenamiento persistente, lo que lo hace más adecuado para proyectos que requieren tanto velocidad en consultas como almacenamiento fiable.

### **Mongo DB**

MongoDB [75] es una base de datos NoSQL de código abierto diseñada para manejar grandes volúmenes de datos de manera eficiente y flexible. En lugar de utilizar el modelo relacional tradicional basado en tablas, MongoDB organiza los datos en documentos JSON o BSON (una representación binaria de JSON), lo que le permite adaptarse a estructuras de datos dinámicas y no estructuradas.Fue seleccionada para la persistencia del historial de chat debido a su modelo flexible y eficiente, que se adapta mejor a las características dinámicas de un sistema de chat en comparación con una base de datos relacional. En este tipo de sistemas, los datos de los mensajes pueden variar en estructura, incluir múltiples campos y crecer rápidamente en volumen, lo que hace que una solución NoSQL como MongoDB sea más adecuada.

El modelo de documentos de MongoDB que permite almacenar cada mensaje como un documento JSON o BSON, ofrece flexibilidad para incluir diferentes atributos sin la necesidad de un esquema rígido, como el requerido en las bases de datos relacionales. Esto facilita agregar nuevas funcionalidades o tipos de datos al sistema sin realizar complejas migraciones de esquema, lo que resulta clave para un sistema de chat que puede evolucionar rápidamente.

### **FastApi**

FastAPI [76] es un framework web desarrollado en Python, diseñado específicamente para facilitar la creación de APIs de alto rendimiento y fácil integración. La elección de FastAPI para este proyecto está directamente relacionada con el hecho de que Python es el lenguaje principal en el que se desarrollará. Python es ampliamente reconocido por su simplicidad, versatilidad y una comunidad robusta, lo que lo convierte en una opción ideal para proyectos  modernos.

El hecho de que FastAPI aproveche al máximo las capacidades de Python, como las anotaciones de tipo, refuerza su utilidad en este contexto. Estas anotaciones no solo mejoran la claridad y el mantenimiento del código, sino que también permiten generar automáticamente validaciones de datos y documentación interactiva, características que son fundamentales para un desarrollo rápido y eficiente.

Además, al ser un framework nativo de Python, FastAPI se integra perfectamente con otras bibliotecas y herramientas del ecosistema Python que se utilizarán en el proyecto, como las relacionadas con la persistencia de datos, procesamiento asíncrono, o incluso machine learning si se requiere en etapas futuras. Esto evita problemas de compatibilidad o configuraciones complejas que podrían surgir al trabajar con frameworks basados en otros lenguajes.

## **Conclusiones parciales**

A partir de la revisión bibliográfica realizada en este capítulo, se puede concluir lo siguiente:

* Los asistentes virtuales de tipo chatbots son agentes conversacionales que permiten el acceso a datos y servicios, así como el intercambio de información mediante un diálogo en lenguaje natural.
* El PLN es la base fundamental de los chatbots, ya que permite que estos sistemas entiendan, interpreten y generen respuestas en lenguaje humano. Mediante técnicas de PLN, los chatbots pueden analizar y responder a entradas de usuario con mayor precisión, abarcando desde preguntas simples hasta interacciones complejas, mejorando así la experiencia de usuario.
* Los LLMs, están diseñados para comprender y generar texto en lenguaje natural. Cuando se integran en chatbots, estos modelos permiten generar respuestas más coherentes y contextualmente adecuadas, basándose en patrones de lenguaje aprendidos.
* La técnica RAG complementa esta capacidad de los LLMs al integrar una búsqueda de información en tiempo real, extrayendo datos relevantes de una base documental para proporcionar respuestas actualizadas o específicas sin necesidad de reentrenar el modelo. Esto mejora la precisión de los chatbots en dominios que requieren datos externos actualizados.
* Los agentes inteligentes en chatbots ayudan a gestionar múltiples tareas y decisiones. Estos agentes no solo responden preguntas, sino que también deciden qué fuentes consultar (búsqueda semántica, bases de datos, o respuestas predefinidas) y en qué orden, para resolver consultas complejas.
* Varias universidades han utilizado con éxito chatbots para mejorar sus servicios bibliotecarios, demostrando el potencial y uso adecuado de esta tecnología.

# **Capítulo 2: Solución Propuesta**

## **2.1. Introducción**

En este capítulo se presenta la solución propuesta para el desarrollo del sistema, documentando de manera detallada los requisitos de software necesarios para su implementación, el diagrama de casos de uso que describe las interacciones principales del sistema con sus actores, y los estilos y patrones arquitectónicos seleccionados para garantizar una estructura robusta y eficiente. Además, se incluyen los patrones de diseño empleados para resolver problemáticas específicas en el desarrollo, así como los principios de diseño aplicados para asegurar un sistema escalable, mantenible y alineado con las mejores prácticas de ingeniería de software.

## **2.2. Captura de requisitos**

A continuación, se encuentran listados los requisitos funcionales y de calidad del módulo del asistente virtual de tipo chatbot.

### **2.2.1. Requisitos funcionales**

RF-1 El sistema debe proveer a los clientes de la biblioteca la capacidad de que busquen libros mediante palabras clave, autores, temas o categorías.

RF-2 El sistema debe proveer a los clientes de la biblioteca la capacidad de ofrecer recomendaciones de libros o recursos en función sus intereses o suscorrespondientes historiales de chat.

RF-3 El sistema debe  proveer a los clientes de la biblioteca la capacidad de responder a consultas sobre servicios de la biblioteca.

RF-4 El sistema debe proveer la capacidad al chatbot de recordar el historial de interacciones y recomendaciones previas.

RF-5 El sistema debe proveer la capacidad de  que el chatbot tenga una personalidad.

RF-6 Cuando se recibe una pregunta el sistema debe procesar el texto en lenguaje natural.

RF-7 Cuando se procese el lenguaje natural el sistema debe recuperar información relevante a la pregunta.

RF-8 Cuando se recupere la información relevante a la pregunta el sistema debe elaborar la respuesta en lenguaje natural.

RF-9 El sistema debe proveer la capacidad de responder a interacciones solo basadas en el contexto de la biblioteca universitaria.

RF-10 El sistema debe proveer la capacidad  de superar las faltas de ortografía en la pregunta del usuario y responder.

RF-11 El sistema debe proveer la capacidad de responder diplomáticamente a posibles comentarios groseros.

RF-12 El sistema debe proveer al Técnico de la biblioteca la capacidad de listar, insertar, modificar y eliminar información contextual de la biblioteca.

RF-13 El sistema debe proveer a los clientes de la biblioteca la capacidad de eliminar el historial de chat.

RF-14 El sistema debe proveer a los clientes de la biblioteca la capacidad de responder a consultas sobre las tesis realizadas en la CUJAE.

### **2.2.2. Requisitos de calidad**

Compatibilidad:

* El sistema debe ser capaz de permitir la interoperabilidad entre el módulo del chatbot y el módulo de gestión de la biblioteca mediante el uso de servicios web RESTful, permitiendo el intercambio eficiente de información en formato JSON para procesar las consultas de los usuarios.

Fiabilidad:

* El sistema debe ser capaz de validar los datos de entrada, antes de ser procesados.

**Restricciones**

Despliegue en los servidores de la CUJAE (hosting propio).

Python como lenguaje de programación.

Chroma DB como almacén vectorial.

Mongo DB como gestor de gase de datos.

Ollama para ejecutar LLM.

## **2.3. Casos de Uso del Sistema**

A continuación, se presentan los artefactos relacionados con los Casos de Uso del Sistema.

### **2.3.1. Actores del sistema**

**Técnico:** Es el encargado de insertar, eliminar , actualizar y obtener toda la información contextual referente a la biblioteca, la cual constituye el dominio específico del chatbot, además es el encargado de enviar las preguntas de un usuario específico hacia el chatbot.

**Cliente:** Es el estudiante o profesor consulta al chatbot para obtener información relevante de la biblioteca en cuanto a información los materiales que dispone y los servicios que ofrece.

### **2.3.2. Diagrama de Casos de Uso del Sistema**

A continuación en la Figura 3 se representa el Diagrama de Casos de Uso del Sistema:

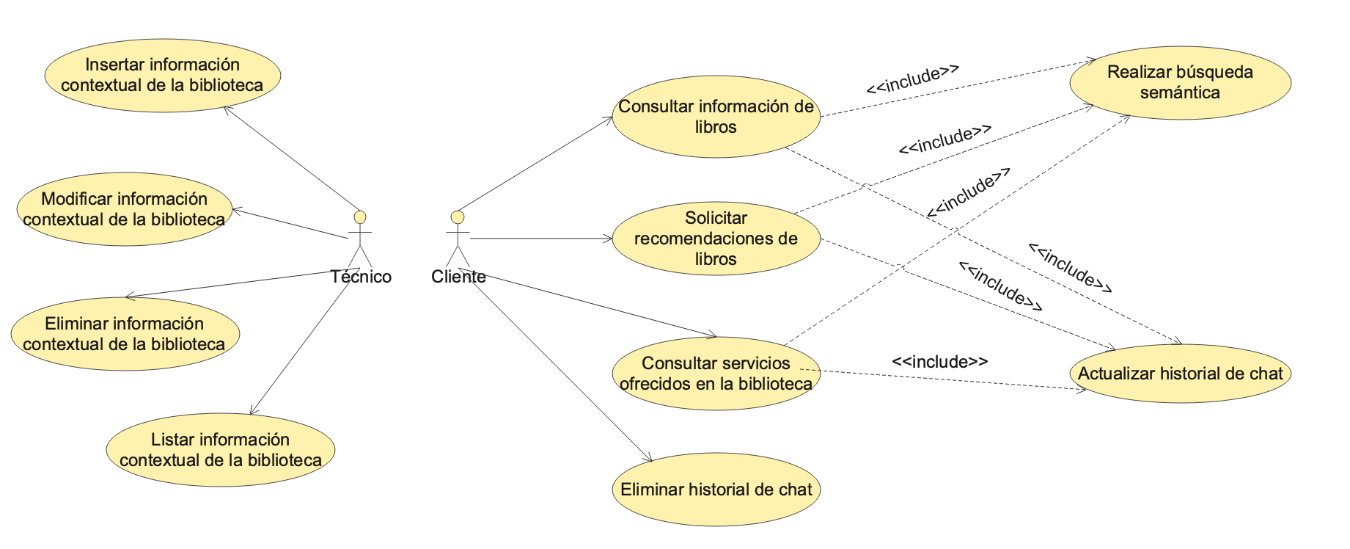


Figura 3: Diagrama de Casos de Uso del Sistema

### **2.3.3. Descripción de los Casos de Uso del Sistema**

Las siguientes tablas muestran la descripción de los casos de uso del sistema:

Tabla 1: Descripción del caso de uso “Insertar información contextual de la biblioteca”

|  |  |
| --- | --- |
| **Caso de uso** | Insertar información contextual de la biblioteca |
| **Actores** | Técnico (inicia) |
| **Descripción** | El caso de uso  inicializa por el Técnico bibliotecario con el Módulo de Gestión cuando inserta información contextual de la biblioteca, la cual puede ser metadatos de los libros existentes o información acerca de los servicios ofrecidos. Luego se almacena de forma vectorial dicha información. El caso de uso concluye cuando se ha insertado la información contextual de forma vectorial. |
| **Requisitos funcionales** | RF-12 |
| **Casos de uso asociados** |  |
| **Precondiciones** | Técnico autenticado en el sistema de la biblioteca |
| **Postcondiciones** | Información contextual de la biblioteca insertada |

Tabla 2: Descripción del caso de uso “Modificar información contextual de la biblioteca”

|  |  |
| --- | --- |
| **Caso de uso** | Modificar información contextual de la biblioteca |
| **Actores** | Técnico (inicia) |
| **Descripción** | El caso de uso  inicializa por el Técnico bibliotecario con   el Módulo de Gestión cuando modifica información contextual de la biblioteca, la cual puede ser metadatos de los libros existentes o información acerca de los servicios ofrecidos. Se actualiza la información almacenada de forma vectorial. El caso de uso concluye cuando se ha modificado la información contextual en el almacén vectorial. |
| **Requisitos funcionales** | RF-12 |
| **Casos de uso asociados** |  |
| **Precondiciones** | Técnico autenticado en el sistema de la biblioteca |
| **Postcondiciones** | Información contextual de la biblioteca modificada |

Tabla 3: Descripción del caso de uso “Eliminar información contextual de la biblioteca”

|  |  |
| --- | --- |
| **Caso de uso** | Eliminar información contextual de la biblioteca |
| **Actores** | Técnico (inicia) |
| **Descripción** | El caso de uso  inicializa por el Técnico bibliotecario con  el Módulo de Gestión cuando elimina información contextual de la biblioteca, la cual puede ser metadatos de los libros existentes o información acerca de los servicios ofrecidos. El caso de uso concluye cuando se ha eliminado  información contextual en el almacén vectorial. |
| **Requisitos funcionales** | RF-12 |
| **Casos de uso asociados** |  |
| **Precondiciones** | Técnico autenticado en el sistema de la biblioteca |
| **Postcondiciones** | Información contextual de la biblioteca eliminada |

Tabla 4: Descripción del caso de uso “Listar información contextual de la biblioteca”

|  |  |
| --- | --- |
| **Caso de uso** | Listar información contextual de la biblioteca |
| **Actores** | Técnico (inicia) |
| **Descripción** | El caso de uso  inicializa por el Técnico bibliotecario con  el Módulo de Gestión cuando solicita listar información contextual de la biblioteca, la cual puede ser metadatos de los libros existentes o información acerca de los servicios ofrecidos. El caso de uso concluye cuando se ha lista la información contextual en el almacén vectorial. |
| **Requisitos funcionales** | RF-12 |
| **Casos de uso asociados** |  |
| **Precondiciones** | - |
| **Postcondiciones** | Información contextual de la biblioteca listada |

Tabla 5: Descripción del caso de uso “Consultar información de libros”

|  |  |
| --- | --- |
| **Caso de uso** | Consultar información de libros |
| **Actores** | Técnico (inicia) |
| **Descripción** | El caso de uso  inicializa por el Técnico bibliotecario con el Módulo de Gestión cuando envía la pregunta de un usuario acerca de un libro. Luego se invoca al caso de uso “Realizar búsqueda semántica” para encontrar la información relativa a la consulta. El caso de uso concluye cuando el chatbot elabora la respuesta para la consulta enviada por el Módulo de Gestión. Finalmente se invoca al caso de uso “Actualizar historial de chat” con la pregunta del usuario recibida desde el Módulo de Gestión y la respuesta del chatbot. |
| **Requisitos funcionales** | RF-1,RF-5,RF-6,RF-8, RF-10,RF-11 |
| **Casos de uso asociados** | Realizar búsqueda semántica (include)  Actualizar historial de chat (include) |
| **Precondiciones** | Técnico autenticado en el sistema de la biblioteca |
| **Postcondiciones** | Respuesta del chatbot elaborada |

Tabla 6: Descripción del caso de uso “Solicitar recomendaciones de libros”

|  |  |
| --- | --- |
| **Caso de uso** | Solicitar recomendaciones de libros |
| **Actores** | Cliente (inicia) |
| **Descripción** | El caso de uso  inicializa por el Cliente con  el Módulo de Gestión cuando envía la solicitud de un usuario de recomendaciones de libros de un tema específico. Luego se invoca al caso de uso “Realizar búsqueda semántica” para encontrar la información relativa a la solicitud. El caso de uso concluye cuando el chatbot elabora la respuesta para la consulta enviada por el Módulo de Gestión. Finalmente se invoca al caso de uso “Actualizar historial de chat” con la pregunta del usuario recibida desde el Módulo de Gestión y la respuesta del chatbot. |
| **Requisitos funcionales** | RF-2,RF-5,RF-6,RF-8, RF-10,RF-11 |
| **Casos de uso asociados** | Realizar búsqueda semántica (include)  Actualizar historial de chat (include) |
| **Precondiciones** | Cliente autenticado en el sistema de la biblioteca |
| **Postcondiciones** | Respuesta del chatbot elaborada |

Tabla 7: Descripción del caso de uso “Consultar servicios ofrecidos en la biblioteca”

|  |  |
| --- | --- |
| **Caso de uso** | Consultar servicios ofrecidos en la biblioteca |
| **Actores** | Módulo de Gestión (inicia) |
| **Descripción** | El caso de uso  inicializa por el Cliente con   el Módulo de Gestión cuando envía la consulta de un usuario acerca de los servicios ofrecidos por la biblioteca. Luego se invoca al caso de uso “Realizar búsqueda semántica” para encontrar la información relativa a la consulta. El caso de uso concluye cuando el chatbot elabora la respuesta para la consulta enviada por el Módulo de Gestión. Finalmente se invoca al caso de uso “Actualizar historial de chat” con la pregunta del usuario recibida desde el Módulo de Gestión y la respuesta del chatbot. |
| **Requisitos funcionales** | RF-3,RF-5,RF-7,RF-8, RF-10,RF-11 |
| **Casos de uso asociados** | Realizar búsqueda semántica (include)  Actualizar historial de chat (include) |
| **Precondiciones** | Cliente autenticado en el sistema |
| **Postcondiciones** | Respuesta del chatbot elaborada |

Tabla 8: Descripción del caso de uso “Realizar búsqueda semántica”

|  |  |
| --- | --- |
| **Caso de uso** | Realizar búsqueda semántica |
| **Actores** | - |
| **Descripción** | El caso de uso inicia una vez conocida la consulta en lenguaje natural de un cliente que ha llegado desde el Módulo de Gestión. El caso de uso concluye cuando se devuelve la información contextual de la biblioteca en orden de similitud con la consulta recibida. |
| **Requisitos funcionales** | RF-7 |
| **Casos de uso asociados** | - |
| **Precondiciones** | Consulta recibida desde el Módulo de Gestión |
| **Postcondiciones** | Información contextual de la biblioteca en orden de similitud con la consulta recibida. |

Tabla 9: Descripción del caso de uso “Actualizar historial del chat”

|  |  |
| --- | --- |
| **Caso de uso** | Actualizar historial de chat |
| **Actores** | - |
| **Descripción** | El caso de uso inicia una vez el chatbot ha elaborado una respuesta ante una consulta. El caso de uso concluye cuando se almacena la consulta recibida del Módulo de Gestión correspondiente a la consulta de un usuario, y cuando el chatbot elabora la respuesta. |
| **Requisitos funcionales** | RF-4,RF-9 |
| **Casos de uso asociados** | - |
| **Precondiciones** | Consulta recibida desde el Módulo de Gestión.  Respuesta elaborada por el chatbot |
| **Postcondiciones** | Historial de chat actualizado |

Tabla 10: Descripción del caso de uso “Eliminar historial del chat”

|  |  |
| --- | --- |
| **Caso de uso** | Eliminarhistorial de chat |
| **Actores** | - |
| **Descripción** | El caso de uso inicia por un cliente cuando elimina su historial de chat . El caso de uso concluye cuando se elimina el historial de chat de el cliente. |
| **Requisitos funcionales** | 13 |
| **Casos de uso asociados** | - |
| **Precondiciones** | Cliente autenticado en el sistema |
| **Postcondiciones** | Historial de chat eliminado |

## **2.4. Arquitectura de la solución propuesta**

A continuación se presenta la estructura general del sistema, incluyendo los elementos principales, su interacción y cómo se organizan para cumplir con los requisitos del proyecto.

### **2.4.1. Estilo arquitectónico: Llamada y Retorno**

Este estilo se presenta en el proyecto mediante el uso de los patrones arquitectónicos: N-Capas y Cliente Servidor.

#### **2.4.1.1. Patrón arquitectónico: N - Capas**

El patrón arquitectónico N - Capas organiza el sistema en niveles jerárquicos, donde cada capa agrupa funcionalidades relacionadas, ofreciendo servicios a la capa superior. Este enfoque estructural permite descomponer las aplicaciones en subtareas que corresponden a diferentes niveles de abstracción, facilitando una distribución jerárquica de roles y responsabilidades. Los roles definen el tipo y forma de interacción entre capas, mientras que las responsabilidades se enfocan en la implementación de funcionalidades específicas. Además, las capas inferiores actúan como servicios núcleo, reutilizables en todo el sistema, mientras que las superiores manejan interacciones de mayor nivel [77]. En el desarrollo de este proyecto para la biblioteca universitaria, se utilizó este patrón con un enfoque de reutilización, asegurando modularidad, escalabilidad y flexibilidad al permitir el reemplazo de capas sin alterar la funcionalidad global del sistema.

La arquitectura fue estructurada en 4 capas con un enfoque basado en la  reutilización:

·  Capa específica.

·  Capa general.

·  Capa intermedia

·  Capa software de sitemas.

En la Figura 4 se presenta el Patrón N - Capas:

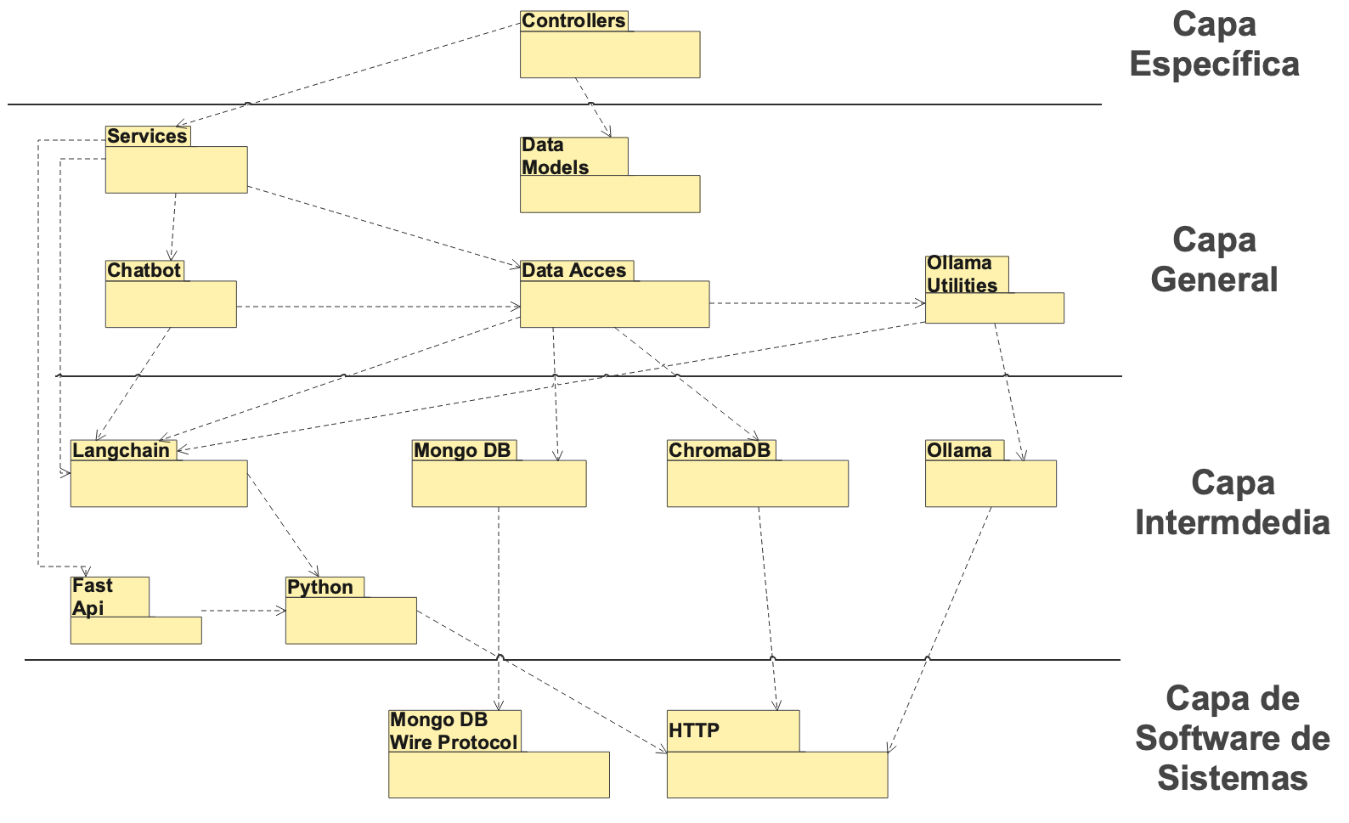


Figura 4: Representación del Patrón Arquitectónico N - Capas

**Capa de software del sistema:** Esta capa incluye los protocolos, componentes y subsistemas provistos por el sistema operativo, formando la base tecnológica sobre la que se desarrolla el resto de las funcionalidades del sistema. En este proyecto, los protocolos identificados son HTTP, que permite la comunicación entre cliente y servidor a través de la web, y el MongoDB Wire Protocol, utilizado para interactuar con la base de datos MongoDB a nivel binario, garantizando un acceso eficiente y estructurado a los datos.

**Capa intermedia:** Aquí se agrupan los paquetes, frameworks y subsistemas que no son desarrollados específicamente para este sistema, sino que provienen de herramientas de terceros. Este nivel aporta soluciones preexistentes para acelerar el desarrollo y mejorar la robustez del sistema. En este caso, incluye tecnologías como Langchain (para manejar cadenas de lenguaje natural), FastAPI (para construir la API de forma eficiente), Ollama (para generación de respuestas basadas en modelos de lenguaje), ChromaDB (para almacenamiento y recuperación de datos vectoriales), Python (como lenguaje de programación principal), y MongoDB (para la persistencia de datos).

**Capa general:** Es la capa donde reside la lógica principal del sistema, compuesta por elementos que pueden ser reutilizados para diversos propósitos dentro del sistema. Aquí se encuentran los servicios, que implementan las reglas del negocio, el chatbot, los mecanismos de acceso a datos, los modelos de datos, que representan la estructura de la información, y las utilidades para interactuar con Ollama, que permiten la integración con este modelo.

**Capa específica:** Esta capa se encarga de exponer la funcionalidad del sistema al usuario mediante controllers. Estos controladores son responsables de definir y gestionar los endpoints, permitiendo que los servicios implementados en la capa general sean accesibles para las solicitudes externas.

#### **2.4.1.2. Patrón arquitectónico: Cliente Servidor**

El sistema implementa el patrón Cliente-Servidor porque establece una clara interacción entre el Módulo de Gestión (cliente) y la API del asistente virtual de tipo chatbot (servidor) [78]. En esta relación, el Módulo de Gestión actúa como cliente al consumir los servicios expuestos por la API del chatbot para cumplir funciones específicas tanto de manejo de información contextual de la biblioteca como de intermediación en las interacciones de los usuarios.En primer lugar, el Módulo de Gestión puede insertar información contextual de la biblioteca, como metadatos de los libros existentes, enviando solicitudes a la API del chatbot. Este último, actuando como servidor, almacena dicha información en Chroma DB, gestionando los datos de manera eficiente y estructurada. De manera similar, el Módulo de Gestión puede solicitar la eliminación, edición o consulta de esa información, delegando el manejo y almacenamiento de los datos al servidor. En segundo lugar, el módulo también juega un papel fundamental en el procesamiento de las preguntas de los usuarios. Cuando un usuario realiza una consulta, el Módulo de Gestión envía la pregunta al chatbot, verificando previamente la autorización del usuario. La API del chatbot, como servidor, procesa la consulta, elabora una respuesta y la devuelve al cliente (Módulo de Gestión), quien finalmente la presenta al usuario. Este enfoque basado en el patrón Cliente-Servidor permite separar las responsabilidades entre el cliente, que realiza tareas de autenticación y control de acceso, y el servidor, que gestiona la lógica principal del sistema, el procesamiento de lenguaje natural y el almacenamiento de datos. Esta arquitectura mejora la modularidad, facilita la escalabilidad del sistema y garantiza un manejo centralizado y eficiente de la información. A continuación en la Figura 5 se representa este patrón arquitectónico.

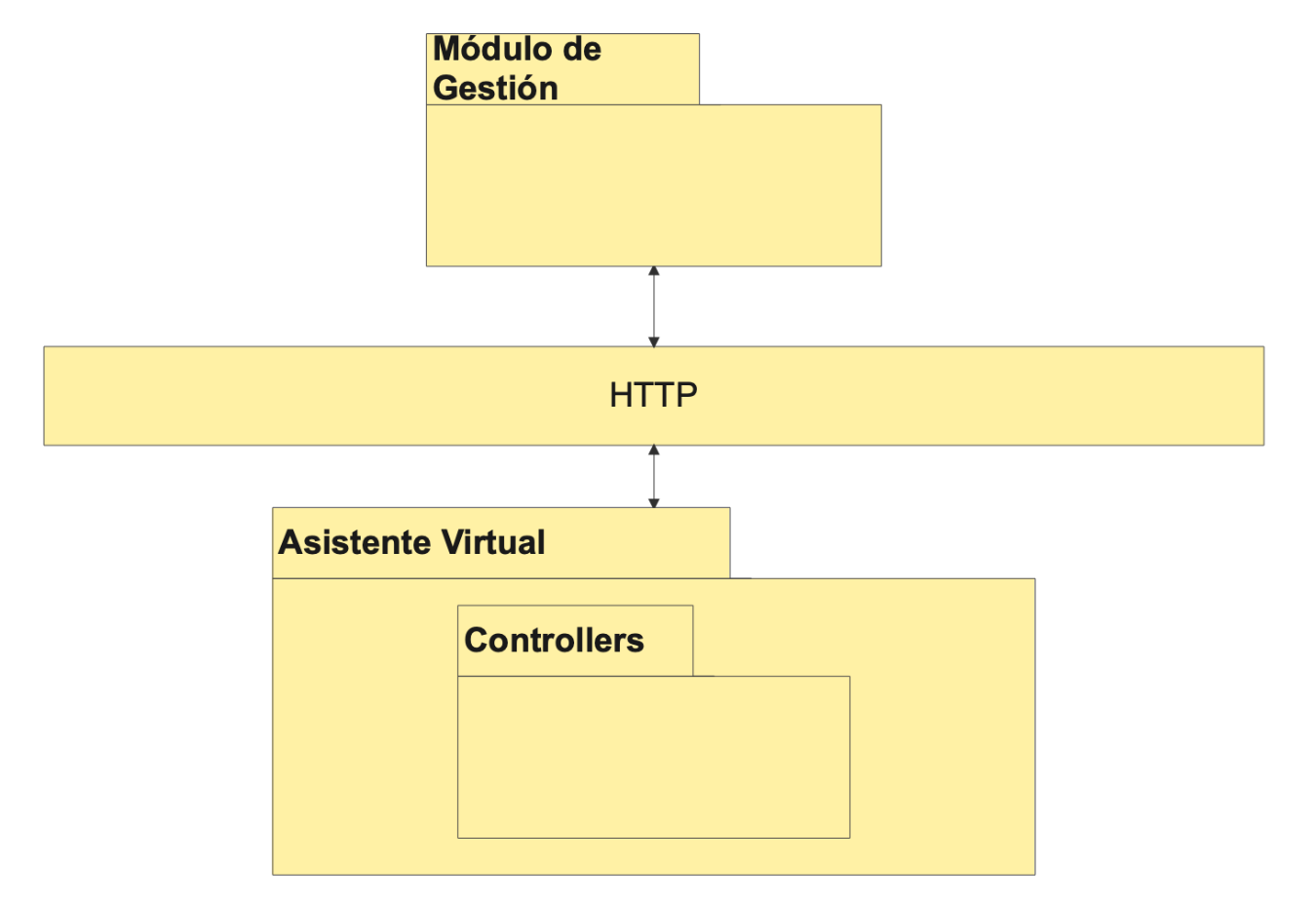


Figura 5: Representación del Patrón Arquitectónico Cliente Servidor

### **2.4.2. Estilo arquitectónico: Centrado en datos**

Este estilo se presenta en el proyecto mediante el uso del patrón arquitectónico: Pizarra.

#### **2.4.2.1. Patrón arquitectónico: Pizarra**

El patrón Pizarra es un modelo arquitectónico de software habitualmente utilizado en sistemas expertos, multiagente y basados en el conocimiento. Consta de múltiples funcionales, denominados agentes, y un instrumento de control denominado pizarra. Los agentes están especializados en resolver una tarea concreta. Todos ellos cooperan para alcanzar una meta común, si bien sus objetivos individuales no están aparentemente coordinados [79]. En la Figura 6 se representa este patrón que es utilizado en el proyecto para representar la interacción colaborativa entre diferentes elementos del sistema.

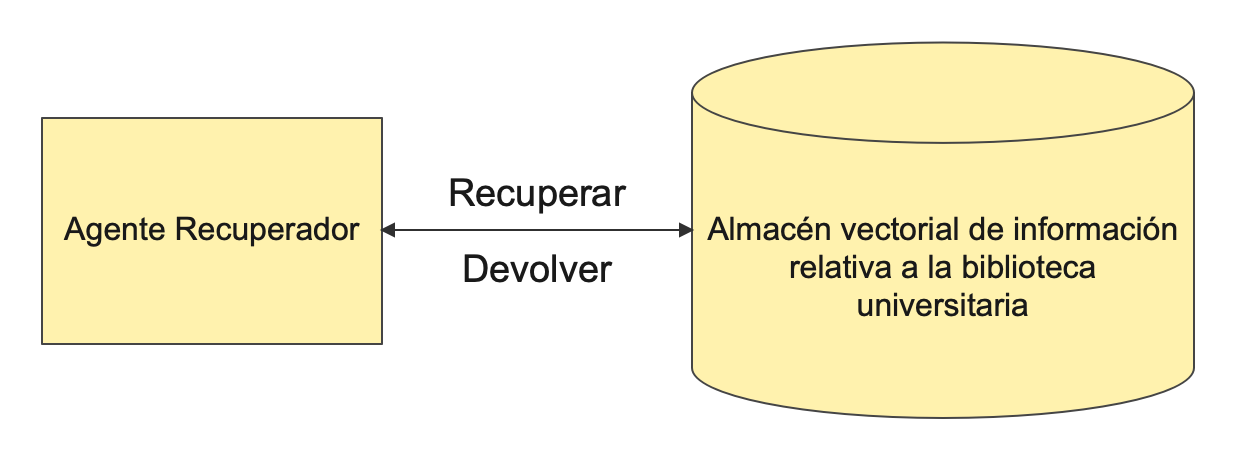


Figura 6: Representación del Patrón Arquitectónico Pizarra

En este proyecto, la **base de datos** actúa como la **pizarra** **central** del sistema, funcionando como un repositorio único donde se almacena y gestiona toda la información que representa el contexto de la biblioteca universitaria. Este repositorio se implementa mediante un almacén vectorial que contiene metadatos sobre los libros existentes en la biblioteca y detalles sobre los servicios ofrecidos.

El **agente recuperador**, un componente esencial del sistema, accede a esta pizarra central para obtener la información necesaria que permite responder a preguntas recibidas por el chatbot. Estas consultas están basadas en el contexto académico y de servicios de la biblioteca, asegurando que las respuestas sean precisas y relevantes para los usuarios.

### **2.4.3. RAG con uso de agente**

El diagrama de la Figura 7 describe el proceso del de RAG utilizando un agente implementado en el chatbot para la biblioteca universitaria. A continuación, se detalla cada paso del proceso:

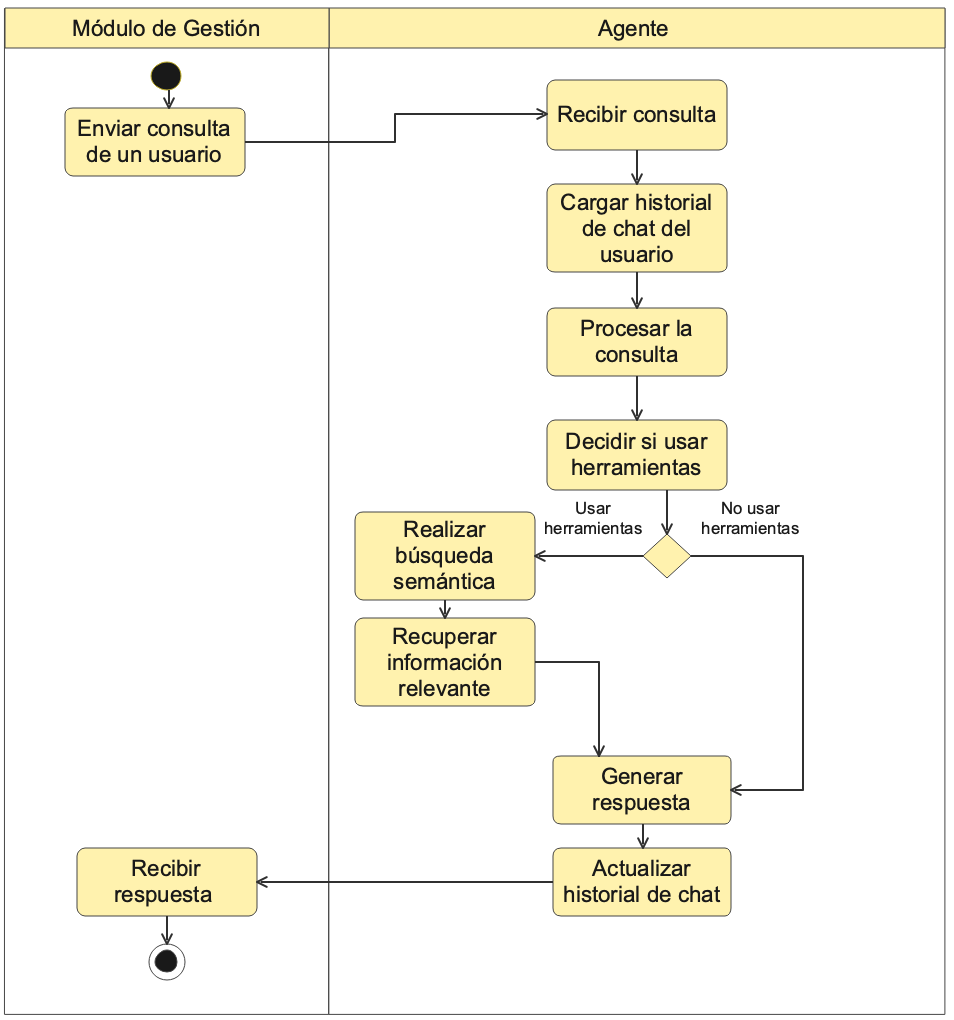


Figura 7: Proceso del de RAG utilizando un agente

**1. Módulo de Gestión**

* **Enviar consulta de un usuario**: El usuario envía su consulta al sistema desde una interfaz de usuario, la recibe el backend del Móodulo de Gestión de la biblioteca y la consulta es enviada al agente que gestiona las respuestas en el Módulo del Chatbot.

**2. Agente**

* **Recibir consulta**: El agente recibe la consulta enviada por el usuario y la comienza a procesar.
* **Cargar historial de chat del usuario**: El agente accede al historial de interacciones del usuario para entender el contexto de la consulta actual.
* **Procesar la consulta**: La consulta es analizada por el agente, que identifica su intención y determina qué tipo de respuesta debe generar.
* **Decidir si usar herramientas**: En este paso, el agente evalúa si es necesario utilizar herramientas externas para proporcionar una respuesta adecuada.
  + **Si no se usan herramientas**: El agente procede directamente a generar una respuesta (basándose en el historial y la información interna).
  + **Si se usan herramientas**: El agente continúa con los siguientes pasos.

**Se usan herramientas cuando:**

* **Realizar búsqueda semántica**: Se lleva a cabo una búsqueda semántica en una base de datos vectorial para encontrar recursos relevantes relacionados con la consulta.
* **Recuperar información relevante**: La información encontrada en la búsqueda es recuperada y preparada para incluirla en la respuesta.
* **Generar respuesta**: El agente genera una respuesta, ya sea basándose en la información recuperada o directamente en la consulta del usuario.
* **Actualizar historial de chat**: El historial del chat del usuario se actualiza con la nueva interacción para mejorar futuras consultas.

**3. Módulo de Gestión**

* **Recibir respuesta**: Finalmente, la respuesta generada por el agente es enviada de regreso al módulo de gestión para ser presentada al usuario.

## **2.5. Diseño e implementación**

A continuación, se presentan los patrones de diseño, que ofrecen soluciones probadas para problemas comunes en la programación. Posteriormente, se describen los principios de diseño, que guían la construcción de sistemas robustos y fáciles de mantener. Finalmente, se abordan los aspectos de implementación, que cubren las decisiones técnicas y herramientas utilizadas en el desarrollo del sistema.

### **2.5.1. Patrones de diseño**

La Figura 8 muestra el Diagrama de Clases UML de las clases utilizadas de Langchain, de la cual se derivan los siguientes patrones de diseño:

**Strategy**

* **También conocido como:** Polimorfismo de comportamiento.
* **Breve descripción del patrón:** es un patrón de diseño de comportamiento que te permite definir una familia de algoritmos, colocar cada uno de ellos en una clase separada y hacer sus objetos intercambiables.
* **Dónde fue utilizado:** En la relación entre Agent y Tool.
* **Por qué fue utilizado:** Para permitir que el agente use diferentes herramientas sin modificar su lógica principal, facilitando la extensibilidad y personalización del comportamiento del agente.

**Command**

* **También conocido como:** Encapsulación de solicitudes.
* **Breve descripción del patrón:** es un patrón de diseño de comportamiento que convierte una solicitud en un objeto independiente que contiene toda la información sobre la solicitud. Esta transformación te permite parametrizar los métodos con diferentes solicitudes, retrasar o poner en cola la ejecución de una solicitud y soportar operaciones que no se pueden realizar.
* **Dónde fue utilizado:** En la clase AgentExecutor al ejecutar una tarea (execute) delegando la acción al Agent.
* **Por qué fue utilizado:** Para desacoplar la solicitud de ejecución de la lógica interna del agente, permitiendo mayor flexibilidad en la gestión de tareas.

**Adapter**

* **También conocido como**: Wrapper (Envoltorio).
* **Breve descripción del patrón:** Es un patrón de diseño estructural que permite la colaboración entre objetos con interfaces incompatibles.
* **Dónde fue utilizado**:
  + En la clase MongoDBChatMessageHistory, que adapta las operaciones genéricas de historial (add\_message, get\_messages) para trabajar con MongoDB.
  + En la clase ChatOllama, que extiende de BaseLanguageModel y adapta las operaciones de generación y predicción para utilizar el modelo Ollama.
* **Por qué fue utilizado**:
  + Para integrar MongoDB como una fuente de almacenamiento sin modificar la interfaz común BaseChatMessageHistory, asegurando que cualquier implementación de historial pueda usarla de manera uniforme.
  + Para permitir que ChatOllama utilice la interfaz de BaseLanguageModel (con atributos como max\_tokens, temperature y métodos como generate y predict) y adapte su funcionalidad al modelo Ollama, manteniendo la consistencia con otras posibles implementaciones de modelos de lenguaje.

Además de estos patrones de diseño se encuentra:

**Singleton** (Figura 9)

* **También conocido como:** Instancia única.
* **Breve descripción del patrón:** Garantiza que una clase tenga solo una instancia y proporciona un punto de acceso global a ella.
* **Dónde fue utilizado:** En la clase ChromaClientSingleton, que administra una única instancia de chromadb.PersistentClient.
* **Por qué fue utilizado:** Para asegurar que el cliente de ChromaDB sea una instancia única compartida en toda la aplicación, evitando la sobrecarga de crear múltiples conexiones y asegurando consistencia en el acceso a los datos.

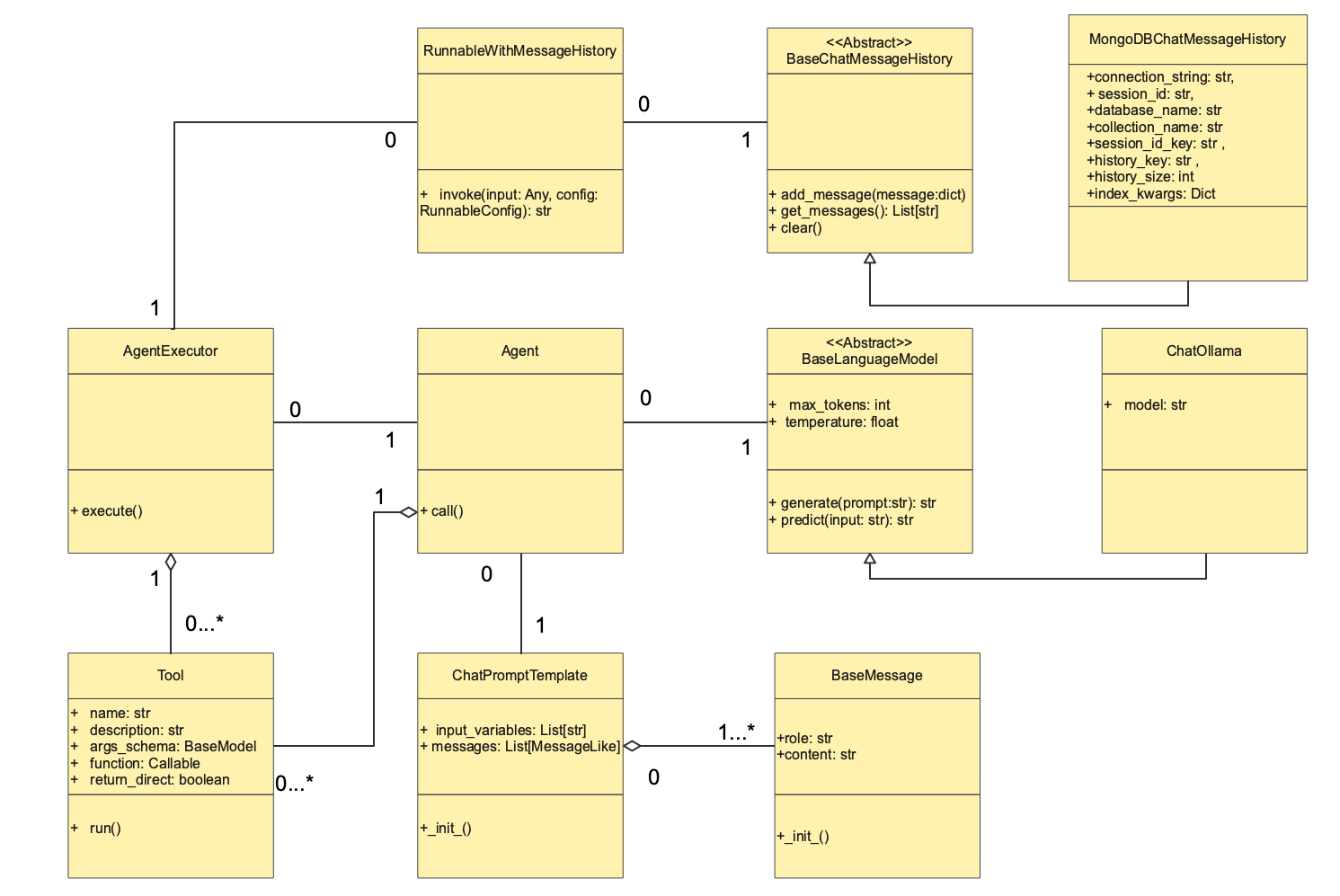


Figura 8: Diagrama de Clases UML de las clases utilizadas de Langchain

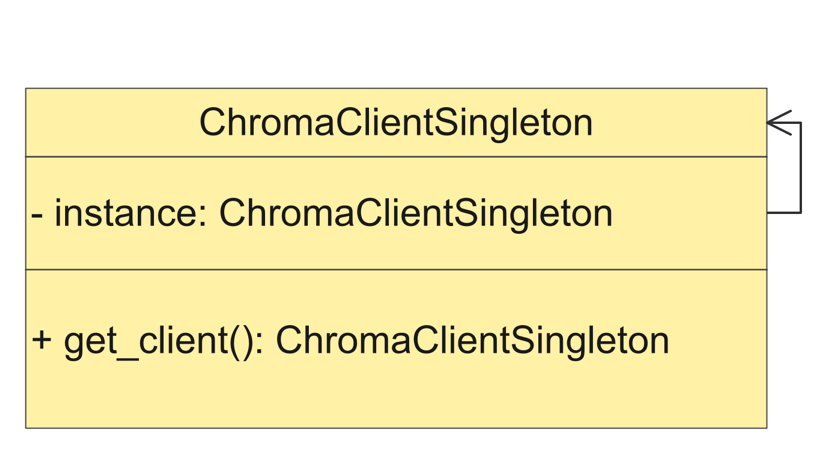


Figura 9: Patrón de diseño Singleton

### **2.5.2. Principios de diseño**

Los principios de diseños utilizados en la solución propuesta son:

* **Principio de Responsabilidad Única**, pues cada clase/fichero tiene una única responsabilidad, se separan las responsabilidades del manejo de solicitudes HTTP, la lógica de negocio y la persistencia de datos en diferentes capas o clases.
* **Principio de Segregación de Interfaces**, puesto que las interfaces son pequeñas y específicas a cada cliente, no se fuerzan dependencias de métodos no utilizados en las implementaciones, y las interfaces definen contratos claros y específicos.
* **Alta cohesión y bajo acoplamiento**, debido a que los módulos tienen alta cohesión (una única responsabilidad), el acoplamiento entre módulos es bajo, y las dependencias entre módulos están bien definidas y minimizadas.
* **Se encapsula la variabilidad**, pues se manejan las configuraciones y opciones variables de manera que están desacopladas del código principal. Ejemplo en el proyecto es la encapsulación del modelo LLM y el modelo para vectorizar, que pueden ser cambiados.

### **2.5.3. Aspectos de implementación**

En este trabajo se utiliza la biblioteca LangChain como marco principal para la construcción de un sistema de chatbot integrado con recuperación de información y memoria contextual. A continuación, se detallan los principales componentes y su propósito dentro de la solución.

Los datos necesarios para las operaciones del chatbot se encuentran almacenados en dos bases de datos vectoriales, definidas en los módulos collection\_\_of\_\_books y collection\_\_of\_\_general\_information. Estas bases de datos se integran mediante herramientas (tools) (Figura 10) específicas que permiten realizar búsquedas eficientes en función de la similitud de los documentos con las consultas del usuario. Por ejemplo, la herramienta get\_results está diseñada para recuperar libros relevantes basándose en temas de interés o búsquedas pasadas del usuario, mientras que get\_library\_information se utiliza para consultar detalles sobre los procesos administrativos de la biblioteca, como el préstamo de libros.

El flujo de conversación está estructurado mediante un esquema de plantillas de mensajes definido con ChatPromptTemplate (Figura 11). Este esquema incluye instrucciones claras sobre el comportamiento del asistente, como responder saludos, reconocer presentaciones del usuario y limitar las respuestas al contexto proporcionado por la base de datos. Para manejar la interacción, el historial del chat se incorpora mediante el marcador de posición MessagesPlaceholder, lo que permite integrar el historial almacenado dinámicamente en las respuestas del agente.

Luego se encuentra la definición del agente principal (Figura 12) a través de create\_tool\_calling\_agent del módulo langgraph.prebuilt. Este agente utiliza un modelo de lenguaje preentrenado (llm), configurado específicamente mediante una implementación personalizada (ollama\_llm) para generar respuestas en lenguaje natural basadas en datos de una biblioteca universitaria. Para manejar la memoria persistente de las interacciones, se emplea RunnableWithMessageHistory, una herramienta que permite almacenar y recuperar el historial de mensajes en una base de datos MongoDB, utilizando MongoDBChatMessageHistory. Esto asegura que el agente pueda mantener contexto entre las interacciones, mejorando la personalización y coherencia del diálogo.

Finalmente, el agente se envuelve en un AgentExecutor, el cual coordina la ejecución de herramientas y la generación de respuestas, con una configuración de verbosidad para facilitar el seguimiento de las operaciones durante las pruebas y el desarrollo.

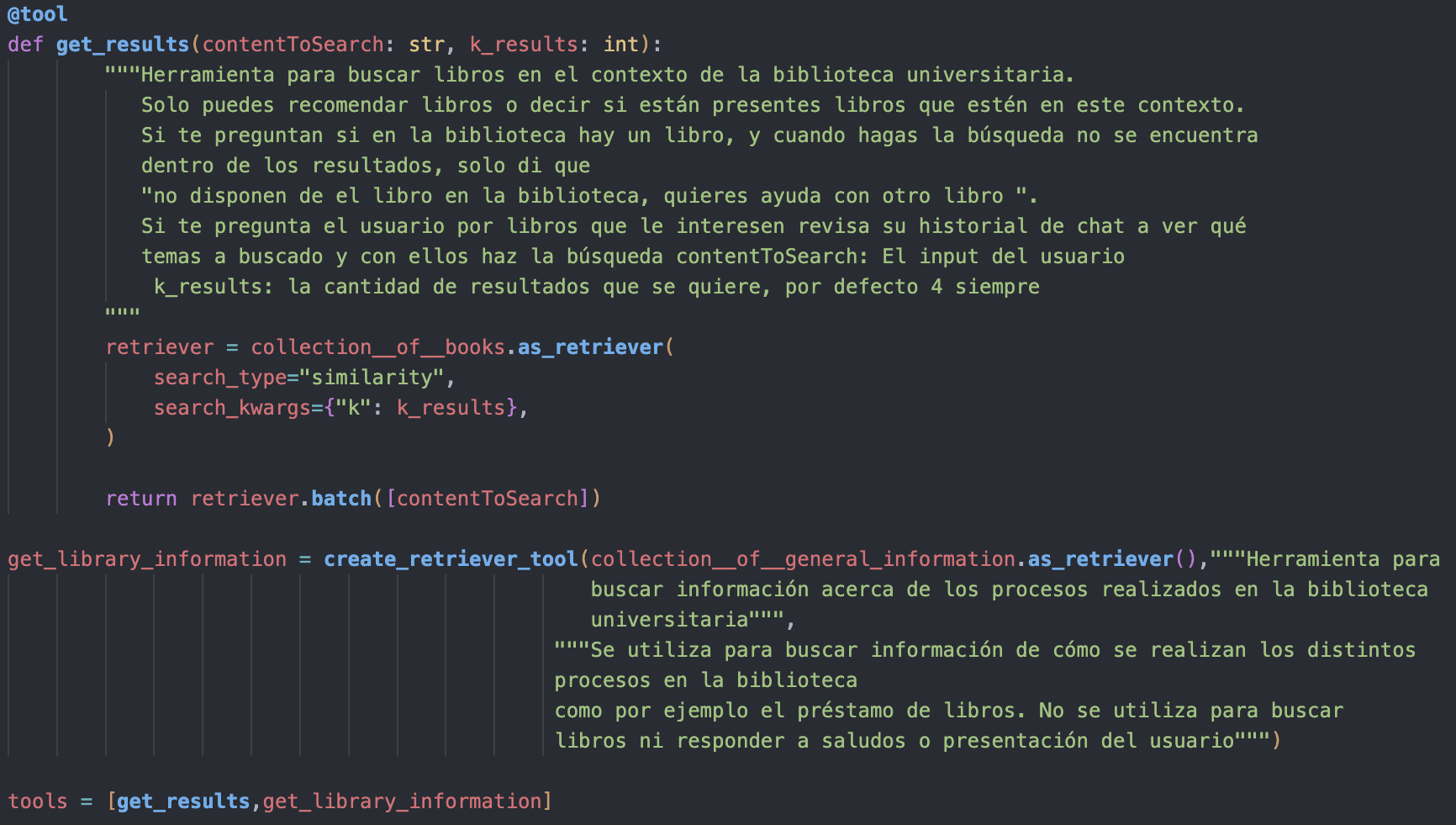


Figura 10: Implementación de herramientas para el agente

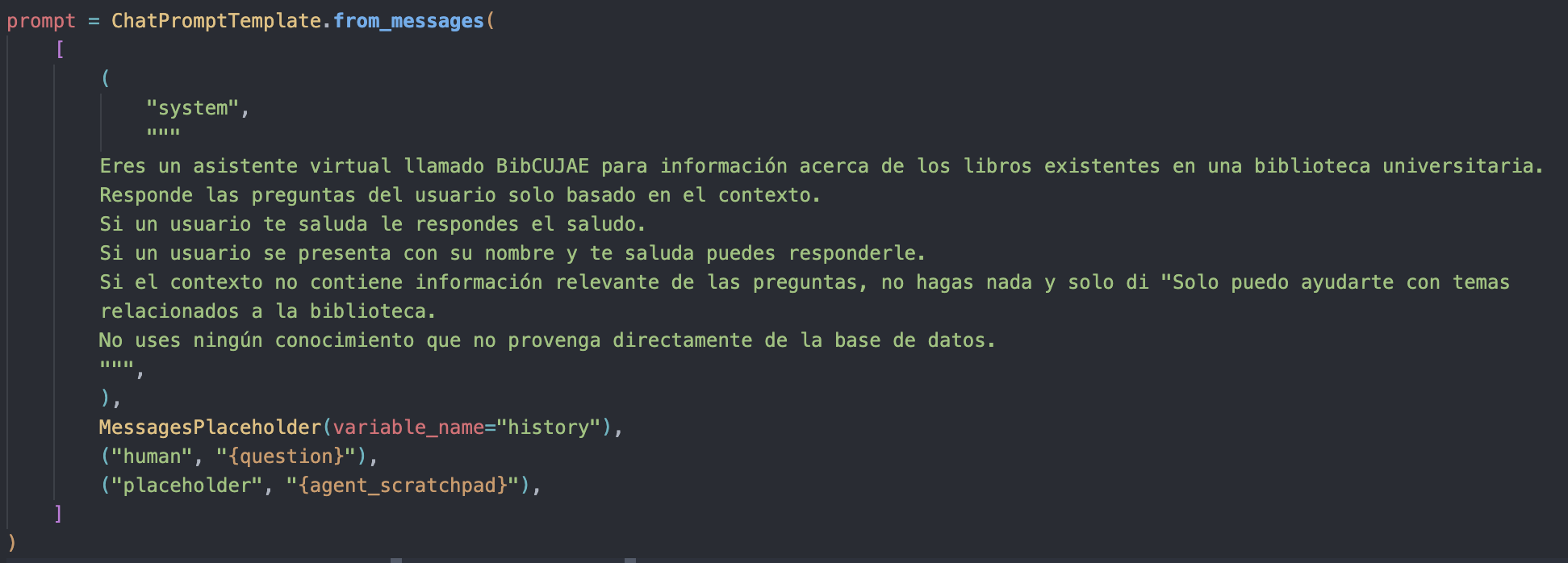


Figura 11: Implementación de la plantilla de chat

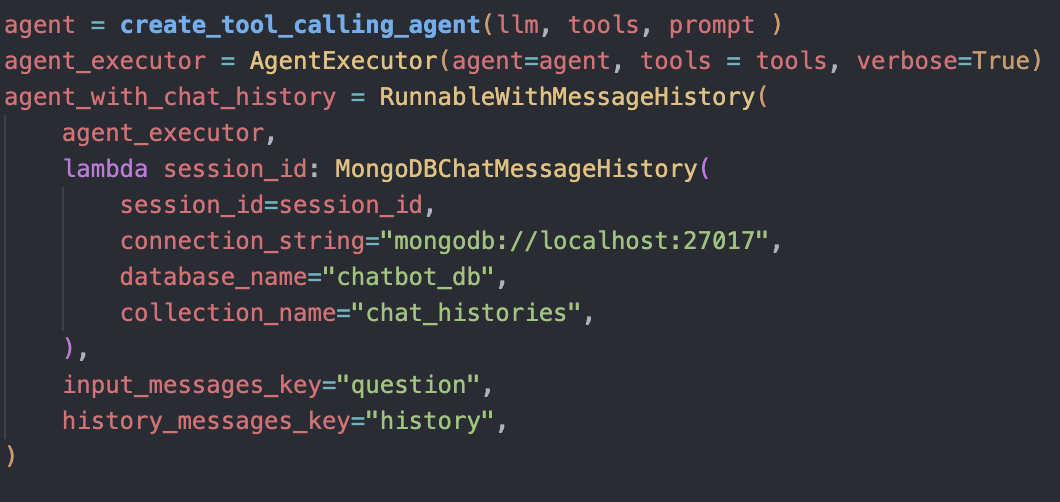


Figura 12: Implementación del agente

## **2.6. Almacenamiento de Datos**

A continuación se presentan los aspectos relativos al almacenamiento de datos en el proyecto:

**ChromaDB**

En el proyecto de la biblioteca, se utiliza Chroma DB para gestionar y almacenar información en formato vectorial, lo que permite realizar búsquedas semánticas y consultas avanzadas. Este sistema de almacenamiento está diseñado para manejar tanto los metadatos de los libros disponibles en la biblioteca como documentos contextuales relacionados con los servicios ofrecidos.

Por un lado, los metadatos de los libros incluyen información clave como el identificador único (ID), título, autor, categoría y año de publicación. Además, se agrega un campo denominado page content, que puede contener el contenido del libro, si está disponible, o una descripción relevante. Esta información se almacena en Chroma DB dentro de una colección específica llamada chroma\_materials, ubicada en una carpeta del proyecto. La vectorización de estos datos permite realizar consultas que van más allá de búsquedas exactas, facilitando recomendaciones y coincidencias semánticas basadas en el contenido.

Por otro lado, para manejar información contextual sobre la biblioteca, como los servicios que ofrece (por ejemplo, cómo funciona el préstamo de libros), se utiliza otra colección llamada chroma\_general\_information. Esta colección almacena documentos que se reciben en formato PDF. A cada documento recibido se le genera un identificador único y se procede a vectorizar su contenido, permitiendo que esta información también sea accesible a través de consultas semánticas. En esta colección, además del contenido vectorizado, se almacena el nombre del documento, proporcionando una referencia clara para identificar cada archivo.

Esta estrategia de almacenamiento en Chroma DB permite centralizar y optimizar la gestión de datos, asegurando que tanto los libros como la información operativa de la biblioteca sean fácilmente accesibles y consultables mediante técnicas avanzadas de recuperación de información.

**MongoDB**

El historial de chat almacenado en MongoDB es utilizado para proporcionar al asistente virtual la capacidad de mantener conversaciones con contexto, logrando una interacción más natural y personalizada con los usuarios. Este enfoque es esencial para cumplir con la función del asistente de responder preguntas relacionadas con los libros y servicios de la biblioteca universitaria, asegurándose de que cada interacción tenga en cuenta el contexto previo y las necesidades específicas de cada usuario.

Se utiliza este historial para:

* **Mantener el contexto de la conversación:** Permite al asistente recordar preguntas y respuestas previas dentro de la misma sesión. Esto es crucial para conversaciones en las que el usuario pueda referirse a temas previamente discutidos, como preguntar por un libro recomendado en una consulta anterior.
* **Persistencia de datos:** Almacenar el historial en MongoDB asegura que las interacciones no se pierdan después de cerrar una sesión. Esto facilita retomar conversaciones pasadas y ofrece continuidad en futuras consultas.
* **Gestión personalizada de usuarios:** El uso de identificadores únicos de sesión (SessionId) permite manejar múltiples usuarios simultáneamente, manteniendo un historial independiente para cada uno. Esto evita confusiones y asegura que cada usuario reciba respuestas adecuadas a su contexto.

Almacenamiento en el historial:

Cada entrada en el historial incluye detalles estructurados que permiten reconstruir el flujo de la conversación. Los principales datos almacenados son:

* **Identificador único:** \_id: Generado automáticamente por MongoDB para identificar cada registro de forma única.
* **Identificador de sesión:** SessionId: Asocia cada mensaje con una sesión específica, permitiendo separar las conversaciones de distintos usuarios.
* **Contenido del historial:** History: Almacena los mensajes, que contienen información detallada como:
  + **Tipo de mensaje:** Indica si el mensaje fue enviado por el usuario (human) o generado por el asistente (ai).
  + **Contenido del mensaje:** El texto real del mensaje (por ejemplo, preguntas del usuario o respuestas del asistente).
  + **Metadatos adicionales:** Información auxiliar, como el estado de las herramientas utilizadas por el agente, o datos sobre el procesamiento de la respuesta.

En la Figura 13 se muestra la colección chat\_histories de la base de datos en MongoDB.

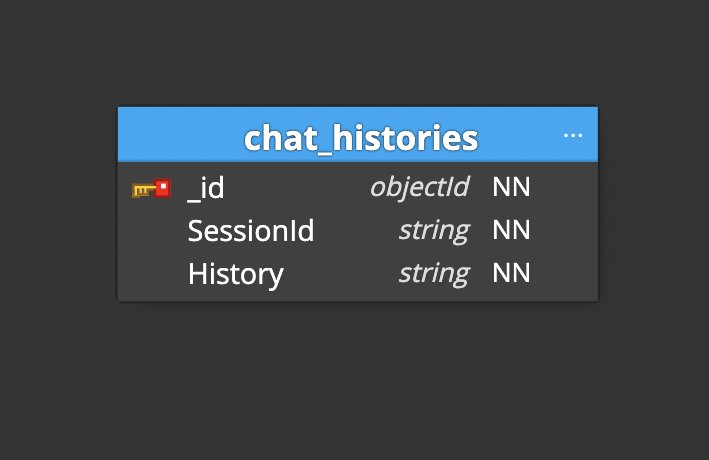


Figura 13: Colección chat\_histories

## **2.7. Modelo de despliegue**

Para garantizar un rendimiento eficiente al desplegar la API que utiliza las herramientas mencionadas (Chroma para bases de datos vectoriales, Ollama con el modelo LLaMA 3.1 versión cuantizada, nomic-embed-text para embeddings, y MongoDB para chats), es necesario considerar cuidadosamente los requisitos de hardware y software.

**Requisitos de Software del nodo**

1. **Sistema Operativo**:
   * **Preferido**: Linux (Ubuntu 22.04 LTS o Debian 11).
   * **Alternativa**: Windows Server 2022, pero Linux suele ser más eficiente y ampliamente soportado para estas tecnologías.
2. **Bases de Datos**:
   * **ChromaDB**: Vector database para embeddings.
   * **MongoDB**: Almacén de datos para gestión de chats.
     + Recomendado usar **MongoDB 6.0** o superior para aprovechar características avanzadas como timeseries.
3. **Modelos de Machine Learning**:
   * **Ollama con LLaMA 3.1 cuantizado**: Configurar Ollama localmente o mediante contenedores Docker.
   * **Nomic-embed-text**: Instalar y configurar las dependencias necesarias para generar embeddings.

**Requisitos de Hardware del nodo**

Dado que se está trabajando con modelos LLM y bases de datos vectoriales, que son intensivos en recursos, los requisitos dependerán del tráfico esperado y la carga de trabajo.

**Requisitos Mínimos  (entorno de pruebas o baja carga):**

* **CPU**: Procesador de 8 núcleos (Intel Xeon o AMD Ryzen).
* **RAM**: 32 GB.
* **GPU**: No es estrictamente necesaria para la versión cuantizada de LLaMA si se despliega en CPU, pero es muy recomendable si el modelo debe responder rápidamente.
* **Almacenamiento**:
  + SSD de 500 GB.
  + Incluye espacio para MongoDB y ChromaDB.

**Requisitos Recomendados (entorno de producción o alta carga):**

* **CPU**: Procesador de 16 núcleos (Intel Xeon Gold o AMD EPYC).
* **RAM**: 64 GB o más, especialmente si se cargan varios embeddings o LLMs en memoria.
* **GPU**: Al menos una GPU NVIDIA con soporte para CUDA, como:
  + NVIDIA A100 o RTX 3090/4090 (24 GB VRAM) para manejar inferencias en tiempo real de LLaMA 3.1.
* **Almacenamiento**:
  + SSD NVMe de 1 TB o más.
  + Separar almacenamiento para bases de datos y cachés temporales.

## **2.8. Conclusiones parciales.**

De forma general se presentan las siguientes conclusiones del capítulo:

* El uso de un agente con herramientas y el enfoque RAG permitió integrar capacidades avanzadas de recuperación de información con generación de respuestas, lo que aseguró que el chatbot pudiera acceder a metadatos de libros, documentos y servicios de la biblioteca para proporcionar respuestas precisas, relevantes y adaptadas a las necesidades de los usuarios.
* El patrón N-Capas permitió estructurar el sistema en módulos independientes y fáciles de mantener, mientras que el patrón Cliente Servidor facilitó la interacción entre el chatbot y el backend para procesar solicitudes en tiempo real.
* La estrategia de almacenamiento en Chroma DB permitió centralizar y optimizar la gestión de datos, asegurando que tanto los libros como la información operativa de la biblioteca sean fácilmente accesibles y consultables mediante técnicas avanzadas de recuperación de información.
* El almacenamiento de datos en MongoDB permitió que el asistente utilice el historial como entrada para generar respuestas más precisas y contextualizadas, además, el facilita la escalabilidad del sistema y su mantenimiento.

# **Capítulo 3: Validación de la solución**

## **3.1. Introducción**

En este capítulo se presenta el proceso de validación del sistema a través de pruebas funcionales, enfocadas en asegurar que el software cumple con los requerimientos funcionales establecidos. Además, se detallan las pruebas realizadas utilizando casos de uso específicos como técnica principal, proporcionando evidencia del correcto funcionamiento del sistema y su alineación con los objetivos planteados.

## **3.2. Pruebas funcionales**

Las pruebas funcionales se enfocan en los requerimientos funcionales del software; permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. Estas pruebas intentan encontrar errores en las categorías siguientes: funciones incorrectas o faltantes, errores de interfaz, errores en las estructuras de datos o en el acceso a bases de datos externas, errores de comportamiento o rendimiento y errores de inicialización y terminación [80].

### **3.2.1. Guiadas por casos de uso**

Al sistema se le realizaron pruebas de caja negra utilizando la técnica guiada por casos de uso que se muestran en las siguientes tablas.

Tabla 11: Caso de Prueba Consultar información  libros

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del caso de prueba** | | Consultar información de libros | | |
| **Caso de uso a probar** | | Consultar información de libros,  Realizar búsqueda semántica,  Actualizar chat | | |
| **Desarrollador** | | Alejandro Estrada Suárez | | |
| **Probador** | | Alejandro Estrada Suárez | | |
| **Objetivo de la prueba** | | Probar la funcionalidad del chatbot que permite consultar información sobre un libro que disponga la biblioteca. | | |
| **Descripción de la prueba** | | La prueba inicia cuando el chatbot recibe una pregunta de un usuario. A continuación, el chatbot procesa la consulta utilizando y accede a la base de datos vectorial  para buscar información relevante sobre el libro solicitado usando una búsqueda semántica. Una vez encontrada la información, el chatbot responde al usuario con los detalles del libro, como su título, autor, disponibilidad y ubicación en la biblioteca. Si no se encuentra información sobre el libro, el chatbot debe proporcionar un mensaje indicando que no se dispone de los datos solicitados y, ayudar con una nueva búsqueda.Finalmente, el historial del chat se actualiza correctamente con la interacción para que pueda ser consultado en futuras sesiones o continuado por el usuario. | | |
| **Condiciones** | | Usuario autenticado en el sistema de la biblioteca | | |
| **CP** | **Valores de entrada** | | **Resultados esperados** | **Resultados reales** |
| 1 | Consulta en lenguaje natural de información acerca de un libro existente:  ¿Qué información hay sobre el libro Tecnología Móvil: Revolución Global ? | | Que el chatbot responda:  “El libro Tecnología Móvil: Revolución Global fue escrito en el 2019 por Raquel Gómez y trata acerca de una guía sobre la evolución de las tecnologías móviles y su impacto en la comunicación global.” | El chatbot respondió:  “El libro "Tecnología Móvil: Revolución Global" fue publicado en 2019 y es una guía sobre la evolución de las tecnologías móviles y su impacto en la comunicación global. Fue escrito por Raquel Gómez.” |
| 2 | Consulta en lenguaje natural de información acerca de un libro no existente:  ¿ Tienen el Libro el Principito en la Biblioteca ? | | Que el chatbot responda:  No tengo información acerca de la existencia de el libro El Principito en nuestra biblioteca. | El chatbot respondió:  Lo siento, pero no tengo información sobre si el Principito está disponible en la biblioteca. Sin embargo, puedo sugerirte buscarlo en nuestra base de datos o consultar con el personal de la biblioteca para obtener más información. |
| **Observaciones** | | | | |
| En el caso de la consulta sobre un libro existente, el chatbot proporcionó una respuesta que, aunque no coincidió textualmente con el resultado esperado, transmitió exactamente la misma información requerida, demostrando precisión en el contenido. Para la consulta sobre un libro no existente, el chatbot respondió adecuadamente indicando que no tenía información disponible y sugiriendo alternativas útiles, como buscar en la base de datos o consultar con el personal de la biblioteca. Aunque la respuesta fue más detallada que el resultado esperado, el mensaje fue claro y relevante. En la Figura 14 se muestra a través de una interfaz de usuario de prueba la interacción con el chatbot y en la Figura 15 desde el editor de código se evidencia cómo el agente utiliza la herramienta get\_results para buscar resultados de libros similares a la consulta del usuario, de los cuales escoge exactamente el que buscaba el usuario. | | | | |

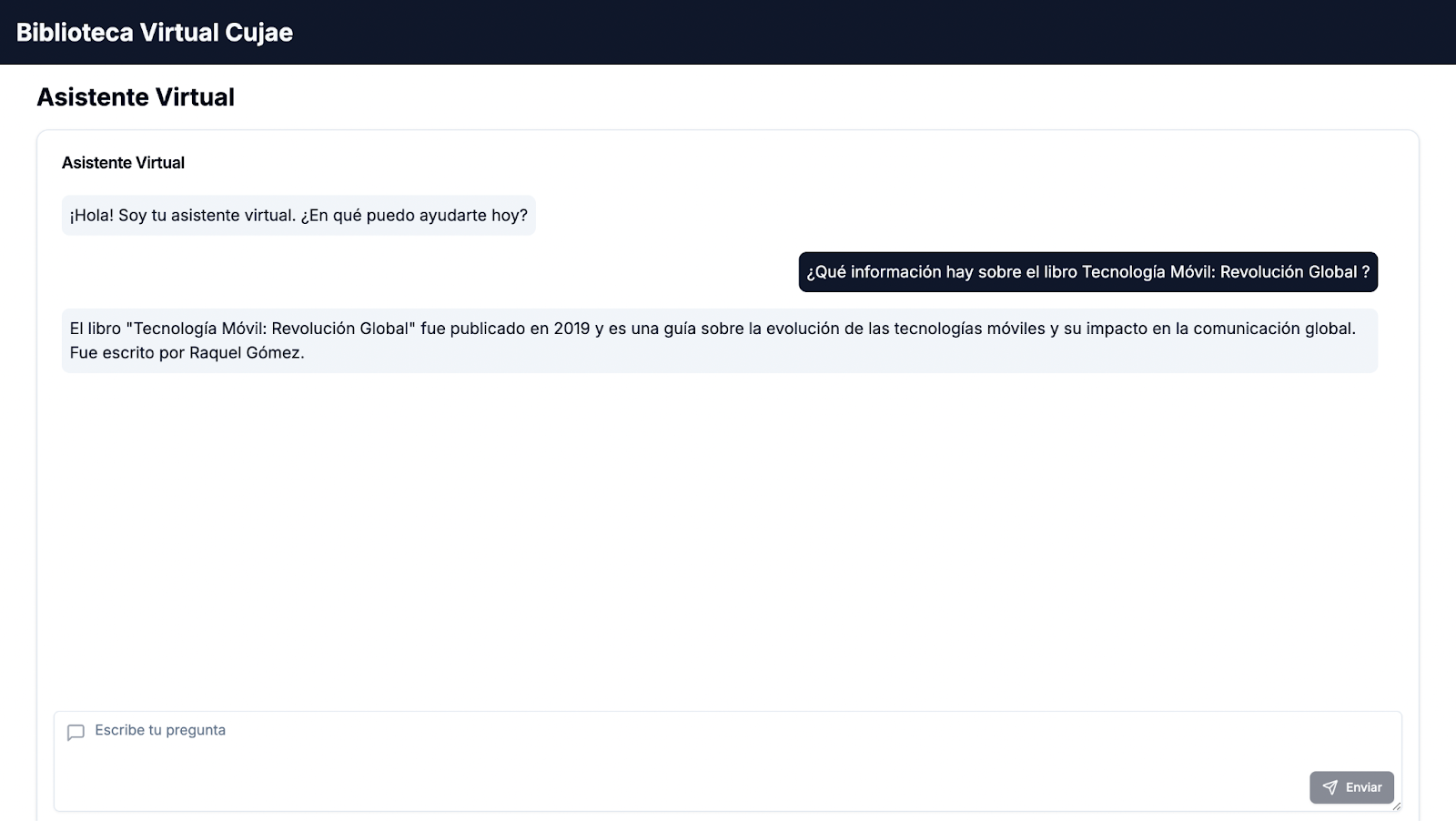


Figura 14: Interacción con el chatbot mediante una interfaz de usuario de prueba para pedir información de un libro



Figura 15: Proceso realizado por el agente para dar respuesta a la consulta del usuario

En la Figura 13 se representa lo sucedido en ante la consulta recibida desde el editor de código, el proceso fue:

1. **Recepción de la Solicitud HTTP**:
   * Se recibió una solicitud HTTP desde el cliente la cual contenía información sobre la sesión (session\_id=001ab) y una entrada de texto codificada en la URL que solicitaba información sobre el libro "Tecnología Móvil: Revolución Global"*.*
2. **Inicio del Proceso en el Agente**:
   * Un nuevo AgentExecutor chain se inicializó para gestionar la solicitud entrante.
   * El agente decide utilizar la herramienta **"get results"**, pasando como entrada el contenido a buscar relacionado con el libro.
3. **Invocación y Recuperación de Resultados**:
   * La invocación de la herramienta get results obtuvo información de la base de datos vectorial que contenía información relevante acerca de los libros existentes.
   * La búsqueda produjo una lista de elementosque coincidían con la categoría de Tecnología*,* incluyendo el libro solicitado y otros relacionados.
4. **Procesamiento de Documentos Recuperados**:
   * Entre los elementos recuperados, se incluyeron los metadatos (autor, categoría, título, año de publicación) y el contenido de la página de cada libro.
   * Se destacó específicamente el libro "Tecnología Móvil: Revolución Global"*,* mencionando que fue publicado en 2019, escrito por Raquel Gómez, y que es una guía sobre la evolución de las tecnologías móviles y su impacto en la comunicación global.
5. **Finalización del Proceso**:
   * La información procesada fue transformada en una respuesta legible para el cliente.
   * El AgentExecutor chain marcó la finalización de su tarea.
6. **Respuesta HTTP al Cliente**:
   * Una nueva solicitud HTTP **GET** fue enviada al cliente desde la misma dirección con el mismo endpoint /chat/. Esta solicitud devolvió un código de estado 200 OK indicando que la operación fue exitosa, junto con la información solicitada.

Tabla 12: Caso de Prueba Solicitar recomendaciones de libros

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del caso de prueba** | | Solicitar recomendaciones de libros | | |
| **Caso de uso a probar** | | Solicitar recomendaciones de libros,  Realizar búsqueda semántica,  Actualizar chat | | |
| **Desarrollador** | | Alejandro Estrada Suárez | | |
| **Probador** | | Alejandro Estrada Suárez | | |
| **Objetivo de la prueba** | | Probar la funcionalidad del chatbot que permite recomendar libros relacionados a temas de interés del usuario. | | |
| **Descripción de la prueba** | | La prueba inicia cuando el chatbot recibe una pregunta de un usuario acerca de recomendación de libros. El chatbot procesa la consulta utilizando su motor de búsqueda semántica y accede a la base de datos vectorial para identificar libros relacionados con los intereses del usuario, basándose en palabras clave, temas o categorías proporcionadas en la consulta o en el historial de chat del usuario.El chatbot genera una lista de recomendaciones, incluyendo detalles como título, autor y breve descripción de cada libro, y la presenta al usuario en el chat.Finalmente, el historial del chat se actualiza correctamente con la interacción para que pueda ser consultado en futuras sesiones o continuado por el usuario. | | |
| **Condiciones** | | Usuario autenticado en el sistema de la biblioteca | | |
| **CP** | **Valores de entrada** | | **Resultados esperados** | **Resultados reales** |
| 1 | Consulta en lenguaje natural de información acerca de recomendación de libros de un tema: Recomiéndame libros de inteligencia artificial | | Que el chatbot responda:  Recomiendo los siguientes libros de inteligencia artificial:  "Inteligencia Artificial y Sociedad" de María López ,  "Creatividad y AI" de Laura García,  "Guerra y AI: La Nueva Frontera" de Ana González,  "Amor en Tiempos de IA" de Carlos Martínez . | El chatbot respondió:  Recomiendo los siguientes libros de inteligencia artificial:  \* "Inteligencia Artificial y Sociedad" de María López (2022)  \* "Creatividad y AI" de Laura García (2021)  \* "Guerra y AI: La Nueva Frontera" de Ana González (2023)  \* "Amor en Tiempos de IA" de Carlos Martínez (2021)  Espero que disfrutes leyendo alguno de estos libros. |
|  |  | |  |  |
| 2 | Consulta en lenguaje natural de información acerca de libros que le interesen (anteriormente se le dijo que “me gustan libros de inteligencia artificial, amor y deporte”):  Recomiéndame libros que me interesen | | Que el chatbot responda:  Libros vinculados a los temas que le interesan al usuario | El chatbot respondió:  Tienes varios libros interesantes relacionados con tu área de interés. Algunos de ellos son:  \* "Amor en Tiempos de IA" de Carlos Martínez, que explora cómo la inteligencia artificial puede afectar las relaciones humanas y el amor en la era digital.  \* "Inteligencia Artificial y Sociedad" de María López, que analiza cómo la inteligencia artificial está transformando nuestra sociedad desde la economía hasta las relaciones interpersonales.  \* "Creatividad y AI" de Laura García, que muestra cómo la inteligencia artificial está influyendo en la creatividad humana y la producción artística.  \* "Psicología Deportiva" de Miguel Torres, que explora la importancia de la psicología en el rendimiento deportivo y cómo los atletas pueden mejorar su enfoque mental.  Espero que esta información te sea útil. |
| **Observaciones** | | | | |
| En la primera consulta, el chatbot proporcionó una lista de recomendaciones relacionadas con el tema solicitado, incluyendo títulos, autores y años de publicación. Aunque agregó información adicional como el año de publicación, esta mejora enriquece la experiencia del usuario sin afectar la claridad o relevancia de la respuesta. La información proporcionada fue precisa y cumplió con las expectativas del caso de prueba.En la segunda consulta, el chatbot generó una lista de libros relacionados con los intereses previamente declarados por el usuario. Aunque la redacción de la respuesta fue más detallada que el resultado esperado, se ajustó correctamente a la intención del caso de prueba al incluir descripciones adicionales que enriquecen la experiencia del usuario.En general, el chatbot cumplió con las expectativas en ambos casos, demostrando capacidad para interpretar la consulta y ofrecer resultados útiles y contextualizados. | | | | |

Tabla 13: Caso de Prueba Consultar servicios de la biblioteca

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del caso de prueba** | | Consultar servicios de la biblioteca | | |
| **Caso de uso a probar** | | Consultar servicios ofrecidos en la biblioteca,  Realizar búsqueda semántica,  Actualizar chat | | |
| **Desarrollador** | | Alejandro Estrada Suárez | | |
| **Probador** | | Alejandro Estrada Suárez | | |
| **Objetivo de la prueba** | | Probar la funcionalidad del chatbot que permite consultar servicios ofrecidos en la biblioteca. | | |
| **Descripción de la prueba** | | La prueba inicia cuando el chatbot recibe una pregunta de un usuario acerca de  cómo funciona un servicio de la biblioteca como por ejemplo los préstamos. El chatbot procesa la consulta utilizando su motor de búsqueda semántica y accede a la base de datos vectorial para identificar la respuesta relevante al usuario. El chatbot genera una respuesta explicando cómo se realiza cierto servicio.Finalmente, el historial del chat se actualiza correctamente con la interacción para que pueda ser consultado en futuras sesiones o continuado por el usuario. | | |
| **Condiciones** | | Usuario autenticado en el sistema de la biblioteca | | |
| **CP** | **Valores de entrada** | | **Resultados esperados** | **Resultados reales** |
| 1 | Consulta en lenguaje natural de información acerca de cómo se realiza el proceso de préstamo en la biblioteca:  ¿Cómo se puede realizar un préstamo en la biblioteca ? | | Para realizar un préstamo en la biblioteca, sigue estos pasos:  1. Solicita al técnico del mostrador tus necesidades informativas (título, autor o materia) y el tipo de documento que deseas (libro, revista o CD).  2. El técnico de la información resolverá tu necesidad informativa utilizando el sistema DOCTUS y el Kardex de revistas.  3. Una vez recuperada la información, el técnico bibliotecario te orientará en el llenado del modelo de solicitud de préstamo con original y copia, revisando que no falte ningún dato para su posible reclamación en caso necesario (datos del usuario y del libro).  4. El modelo de solicitud de préstamo se envía al almacén, a la sala especializada o a Referencia según corresponda para su localización, dejando en el lugar del documento el modelo original y enviando al mostrador el documento con la copia.  5. Según el tipo de documento y la categoría del usuario, se realiza el préstamo del documento:  Si es un libro ejemplar único, el préstamo es interno para los estudiantes y para los profesores solo por una semana si no es demandado.  Si el documento tiene más de un ejemplar, se realizará el préstamo externo al estudiante por 15 días con derecho a renovar sino es demandado. Los profesores tienen igual condición.  6. El no cumplimiento por parte del usuario en la devolución del documento en la fecha indicada ocasionará la imposición de una sanción que limita el préstamo externo según el tiempo que indique el técnico bibliotecario.  7. Si es un documento de Salas Especializadas o Referencia, el préstamo de estos documentos es interno para los estudiantes y si es un profesor se le presta por una semana con derecho a renovación según su demanda. | Para realizar un préstamo en la biblioteca, sigue estos pasos:  1. Solicita al técnico del mostrador tus necesidades informativas (título, autor o materia) y el tipo de documento que deseas (libro, revista o CD).  2. El técnico de la información resolverá tu necesidad informativa utilizando el sistema DOCTUS y el Kardex de revistas.  3. Una vez recuperada la información, el técnico bibliotecario te orientará en el llenado del modelo de solicitud de préstamo con original y copia, revisando que no falte ningún dato para su posible reclamación en caso necesario (datos del usuario y del libro).  4. El modelo de solicitud de préstamo se envía al almacén, a la sala especializada o a Referencia según corresponda para su localización, dejando en el lugar del documento el modelo original y enviando al mostrador el documento con la copia.  5. Según el tipo de documento y la categoría del usuario, se realiza el préstamo del documento:  \* Si es un libro ejemplar único, el préstamo es interno para los estudiantes y para los profesores solo por una semana si no es demandado.  \* Si el documento tiene más de un ejemplar, se realizará el préstamo externo al estudiante por 15 días con derecho a renovar sino es demandado. Los profesores tienen igual condición.  6. El no cumplimiento por parte del usuario en la devolución del documento en la fecha indicada ocasionará la imposición de una sanción que limita el préstamo externo según el tiempo que indique el técnico bibliotecario.  7. Si es un documento de Salas Especializadas o Referencia, el préstamo de estos documentos es interno para los estudiantes y si es un profesor se le presta por una semana con derecho a renovación según su demanda.  Espero que esta información te sea útil. |
|  |  | |  |  |
| **Observaciones** | | | | |
| El chatbot cumplió con los resultados esperados al proporcionar una respuesta detallada y estructurada sobre el proceso de préstamo en la biblioteca. La información proporcionada fue completa, incluyendo todos los pasos necesarios y las condiciones asociadas a los distintos tipos de documentos y categorías de usuarios. Esto indica que el motor de búsqueda semántica y la base de datos vectorial están funcionando correctamente para responder a consultas específicas sobre los servicios de la biblioteca.Además, la respuesta del chatbot coincidió completamente con los resultados esperados, mostrando una gran precisión en la recuperación y presentación de la información. Esto demuestra la capacidad del sistema para interpretar consultas en lenguaje natural y responder de manera útil y consistente. No se identificaron diferencias o errores entre los resultados esperados y los reales, lo que indica un desempeño óptimo de esta funcionalidad en el caso de prueba. | | | | |

Tabla 14: Caso de Prueba Eliminar chat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del caso de prueba** | | Eliminar chat | | |
| **Caso de uso a probar** | | Eliminar historial de chat | | |
| **Desarrollador** | | Alejandro Estrada Suárez | | |
| **Probador** | | Alejandro Estrada Suárez | | |
| **Objetivo de la prueba** | | Probar la funcionalidad que permite que un usuario elimine su historial de chat. | | |
| **Descripción de la prueba** | | La prueba inicia cuando el usuario solicita eliminar su historial de chat. El sistema recibe la solicitud y verifica la sesión activa del usuario para identificar el historial asociado. El sistema procede a eliminar todos los mensajes almacenados en la base de datos relacionados con el identificador de la sesión correspondiente. | | |
| **Condiciones** | | Usuario autenticado en el sistema de la biblioteca | | |
| **CP** | **Valores de entrada** | | **Resultados esperados** | **Resultados reales** |
| 1 | Identificador de sesión de un usuario del sistema : “001ab” | | "message": "chat history deleted successfully" | "message": "chat history deleted successfully" |
| **Observaciones** | | | | |
| La funcionalidad para eliminar el historial de chat cumplió con los resultados esperados. Al proporcionar un identificador de sesión válido, el sistema respondió con el mensaje "chat history deleted successfully", confirmando que el historial asociado fue eliminado correctamente de la base de datos. | | | | |

Tabla 15: Caso de Prueba Insertar información contextual

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del caso de prueba** | | Insertar información contextual | | |
| **Caso de uso a probar** | | Insertar información contextual de la biblioteca | | |
| **Desarrollador** | | Alejandro Estrada Suárez | | |
| **Probador** | | Alejandro Estrada Suárez | | |
| **Objetivo de la prueba** | | Probar la funcionalidad que permite la inserción de información contextual sobre la biblioteca, que sería la base de conocimiento del chatbot. | | |
| **Descripción de la prueba** | | La prueba inicia cuando el sistema recibe una solicitud para insertar información contextual sobre la biblioteca, ya sea mediante metadatos de libros (título, autor, género, etc.) enviados por un endpoint específico o documentos PDF relacionados con los servicios de la biblioteca procesados por otro endpoint. El sistema valida los datos recibidos, extrae el contenido en caso de PDFs, y los almacena como parte de la base de conocimiento. Finalmente, se verifica que la información insertada esté disponible para consultas del chatbot. | | |
| **Condiciones** | |  | | |
| **CP** | **Valores de entrada** | | **Resultados esperados** | **Resultados reales** |
| 1 | Solicitud de inserción de metadatos de libros :  [{      "id": "77", "metadata": { "author": "José Martí","category": "Literatura Infantil",        "title": "La Edad de Oro","year\_of\_publication": 1889      },   "page\_content": "Una revista para niños y niñas, escrita por José Martí, con el objetivo de educar y entretener a los más jóvenes."  }] | | Inserción exitosa. Metadatos vectorizados.  Id: 77 | Inserción exitosa. Metadatos vectorizados.  Id: 77 |
| 2 | Solicitud de inserción de documento pdf con información de los servicios ofrecidos por la biblioteca:  pdf: Procesos realizados en la biblioteca universitaria | | Inserción exitosa. Información de servicios de la biblioteca vectorizados.  Identificador del documento insertado | Identificador del documento insertado: "dbd751c6-16ca-4c24-be13-d0d6e9f7177e  " |
| **Observaciones** | | | | |
| La funcionalidad de inserción de información contextual fue probada exitosamente. En el caso de los metadatos del libro "La Edad de Oro", el sistema permitió su inserción correcta, vectorizando el contenido y asignando el identificador 77, asegurando su disponibilidad para futuras consultas. Para la inserción de un documento PDF sobre los servicios de la biblioteca, el sistema procesó la información adecuadamente y generó un identificador único (dbd751c6-16ca-4c24-be13-d0d6e9f7177e). | | | | |

Tabla 16: Caso de Prueba Modificar información contextual

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del caso de prueba** | | Modificar información contextual | | |
| **Caso de uso a probar** | | Modificar información contextual de la biblioteca | | |
| **Desarrollador** | | Alejandro Estrada Suárez | | |
| **Probador** | | Alejandro Estrada Suárez | | |
| **Objetivo de la prueba** | | Probar la funcionalidad que permite la modificación de información contextual sobre la biblioteca. | | |
| **Descripción de la prueba** | | La prueba inicia cuando el sistema recibe una solicitud para modificar información contextual previamente almacenada sobre la biblioteca relacionada con metadatos de libros . El sistema identifica el contenido a modificar, verifica la validez de los nuevos datos enviados a través del endpoint correspondiente y actualiza la base de conocimiento en consecuencia. | | |
| **Condiciones** | |  | | |
| **CP** | **Valores de entrada** | | **Resultados esperados** | **Resultados reales** |
| 1 | Solicitud de modificación de metadatos de libros existentes:  identificador del libro: 21  Metadatos: author : Patricia Moreno Pérez  (se le actualiza el nombre de la autora agregando el apellido Pérez ) | | El libro con id: 21 con el metadato de author con Patricia Moreno Pérez | El libro con id: 21 con el metadato de author con Patricia Moreno Pérez |
| **Observaciones** | | | | |
| La funcionalidad de modificación de información contextual sobre la biblioteca cumplió con los resultados esperados. En el caso de prueba, el sistema actualizó correctamente el metadato solicitado (autor del libro) y devolvió la respuesta adecuada. | | | | |

Tabla 17: Caso de Prueba Eliminar información contextual

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del caso de prueba** | | Eliminar  información contextual | | |
| **Caso de uso a probar** | | Eliminar  información contextual de la biblioteca | | |
| **Desarrollador** | | Alejandro Estrada Suárez | | |
| **Probador** | | Alejandro Estrada Suárez | | |
| **Objetivo de la prueba** | | Probar la funcionalidad que permite la eliminación de información contextual sobre la biblioteca. | | |
| **Descripción de la prueba** | | La prueba inicia cuando el sistema recibe una solicitud para eliminar información contextual sobre la biblioteca, ya sean metadatos de libros (título, autor, género, etc.) o  documentos relacionados con los servicios de la biblioteca procesados por otro endpoint. Dado los identificadores el sistema procede a realizar la eliminación de la base de datos vectorial | | |
| **Condiciones** | |  | | |
| **CP** | **Valores de entrada** | | **Resultados esperados** | **Resultados reales** |
| 1 | Solicitud de eliminación de metadatos de libros existentes:  identificador: | | "message": "metadata deleted successfully" | "message": "metadata deleted successfully" |
| 2 | Solicitud de eliminación de documento con información de los servicios ofrecidos por la biblioteca:  Identificador: dbd751c6-16ca-4c24-be13-d0d6e9f7177e | | "message": "document deleted successfully" | "message": "document deleted successfully" |
| **Observaciones** | | | | |
| La funcionalidad para eliminar información contextual de la biblioteca cumplió con los resultados esperados en ambos casos de prueba. La eliminación de metadatos de libros y documentos relacionados con los servicios de la biblioteca se realizó correctamente, y el sistema devolvió los mensajes de confirmación esperados. No se identificaron inconsistencias ni errores en la ejecución. | | | | |

Tabla 18: Caso de Prueba Listar información contextual

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del caso de prueba** | | Listar  información contextual | | |
| **Caso de uso a probar** | | Listar  información contextual de la biblioteca | | |
| **Desarrollador** | | Alejandro Estrada Suárez | | |
| **Probador** | | Alejandro Estrada Suárez | | |
| **Objetivo de la prueba** | | Probar la funcionalidad que permite listar información contextual sobre la biblioteca. | | |
| **Descripción de la prueba** | | La prueba inicia cuando el sistema recibe una solicitud para listar información contextual sobre la biblioteca, ya sean metadatos de libros (título, autor, género, etc.) o  documentos relacionados con los servicios de la biblioteca procesados por otro endpoint. Dado los identificadores el sistema procede a listar la información correspondiente | | |
| **Condiciones** | |  | | |
| **CP** | **Valores de entrada** | | **Resultados esperados** | **Resultados reales** |
| 1 | Solicitud de listar  metadatos de libros existentes: | | Listado de la información | Listado de la información |
| 2 | Solicitud de listar  documentos con información de los servicios ofrecidos por la biblioteca: | | Listado de la información | Listado de la información |
| **Observaciones** | | | | |
| La funcionalidad para listar información contextual sobre la biblioteca cumplió con los resultados esperados en ambos casos de prueba, mostrando correctamente los metadatos de libros y los documentos relacionados con los servicios ofrecidos. Esto confirma que el sistema accede y recupera los datos necesarios de forma eficiente. No se detectaron discrepancias entre los resultados esperados y los reales, lo que evidencia que la funcionalidad está operativa. | | | | |

## **3.3. Conclusiones parciales**

De forma general se presentan las siguientes conclusiones del capítulo:

* El sistema respondió de manera precisa a las solicitudes de consulta sobre libros específicos, proporcionando detalles completos como título, autor, categoría y contenido relevante.
* Se comprobó que el sistema genera recomendaciones basadas en las preferencias del usuario y el análisis de metadatos, ofreciendo opciones relevantes y personalizadas.
* Las pruebas demostraron que el sistema puede recuperar y presentar información detallada sobre los servicios de la biblioteca, ayudando a los usuarios a resolver consultas específicas.
* Las pruebas demostraron que el sistema es capaz de recibir, validar, procesar y almacenar tanto metadatos de libros como documentos relacionados con los servicios de la biblioteca, asegurando su disponibilidad para consultas futuras.

# **Conclusiones generales**

De forma general se presentan las siguientes conclusiones:

* Los chatbots, basados en técnicas de PLN y LLMs, facilitan interacciones en lenguaje humano, mejorando la experiencia del usuario mediante respuestas precisas y contextualmente adecuadas.
* El enfoque RAG y el uso de agentes inteligentes optimizan el acceso a información actualizada y la gestión de consultas complejas, destacando su eficacia en entornos como servicios bibliotecarios universitarios.
* El uso de un agente con herramientas y el enfoque RAG permitió que el chatbot proporcionara respuestas precisas y relevantes, mientras que el patrón N-Capas y el patrón Cliente-Servidor mejoraron la estructura del sistema.
* El almacenamiento en Chroma DB optimizó la gestión de datos, y el uso de MongoDB facilitó la generación de respuestas más contextualizadas, mejorando la escalabilidad y el mantenimiento del sistema.
* El sistema demostró ser preciso al responder consultas sobre libros y generar recomendaciones personalizadas basadas en las preferencias del usuario y metadatos.
* Las pruebas confirmaron que el sistema puede recuperar información detallada sobre los servicios de la biblioteca y manejar metadatos y documentos, asegurando su disponibilidad para futuras consultas.

# **Recomendaciones**

Se presentan las siguientes recomendaciones:

* **Ampliar el Soporte Multilingüe**: Probar soporte para varios idiomas, permitiendo que estudiantes internacionales puedan interactuar con el chatbot en su idioma nativo.
* **Integración con el Repositorio de Tesis de la Universidad**: Ampliar las capacidades del chatbot integrándolo con el repositorio de tesis de la universidad, para que pueda proporcionar acceso a estos documentos y ayudar a los estudiantes a buscar tesis relacionadas con su área de interés o tema de investigación.
* **Estandarización en el Estilo de Respuestas**: Asegurarse de que el chatbot tenga un estilo uniforme al generar respuestas. Por ejemplo estandarizar el uso de comillas para los títulos de libros y garantizar que la redacción de las respuestas sea consistente en todos los casos, mejorando la experiencia del usuario y la calidad de la información proporcionada.

# **Referencias Bibliográficas**

[1] S. J. Russell y P. Norvig, *Inteligencia artificial: un enfoque moderno*. Pearson Educación, 2004.

[2] *ARTIFICIAL INTELLIGENCE AND THE DESIGN OF EXPERT SYSTEMS by Luger, George F. & William A. Stubblefield: Hardcover (1989) First Edition; First Printing. | Rare Book Cellar*. Accedido: 4 de marzo de 2024. [En línea]. Disponible en: https://www.abebooks.com/first-edition/ARTIFICIAL-INTELLIGENCE-DESIGN-EXPERT-SYSTEMS-Luger/8646951081/bd

[3] M. A. Boden, *Inteligencia Artificial*. Turner, 2017.

[4] «An experimental study of public trust in AI chatbots in the public sector | Request PDF». Accedido: 21 de octubre de 2024. [En línea]. Disponible en: https://www.researchgate.net/publication/343736024\_An\_experimental\_study\_of\_public\_trust\_in\_AI\_chatbots\_in\_the\_public\_sector

[5] «A Typology of Chatbots in Public Service Delivery | IEEE Journals & Magazine | IEEE Xplore». Accedido: 21 de octubre de 2024. [En línea]. Disponible en: https://ieeexplore.ieee.org/abstract/document/9405373

[6] O. León-Granizo y M. León-Granizo, «Desarrollo de un asistente virtual (chatbot) para mejorar el acceso a la información recurrente por los estudiantes de Instituciones de Educación Superior», *Ecuadorian Sci. J.*, vol. 4, n.o 2, pp. 111-116, sep. 2020, doi: 10.46480/esj.4.2.49.

[7] P. R. Hernandez y D. V. Cruz, «Los Asistentes virtuales basados en Inteligencia Artificial», *ReCIBE Rev. Electrónica Comput. Informática Bioméd. Electrónica*, vol. 11, n.o 2, Art. n.o 2, nov. 2022, doi: 10.32870/recibe.v11i2.251.

[8] R. F. Pesantez Campoverde, «Desarrollo de un sistema e-shop online integrando un chatbot para tomar decisiones utilizando un modelo de lenguaje natural», bachelorThesis, Machala; Universidad Técnica de Machala, 2024. Accedido: 21 de octubre de 2024. [En línea]. Disponible en: http://repositorio.utmachala.edu.ec/handle/48000/23112

[9] J. Consuegra Pérez, «Chatbot/Asistente virtual para mejorar la productividad de las tiendas (2024)», *Área Videojuegos-Tecnol. TFM*, jul. 2024, [En línea]. Disponible en: https://sciencevalue.udit.es/tfm\_videojuegos\_tecnologia/5

[10] S. Pinazo-Hernandis, «Las personas mayores, las tecnologías y los cuidados. Avances y retos», *SCIO Rev. Filos.*, n.o 26, Art. n.o 26, jul. 2024, doi: 10.46583/scio\_2024.26.1152.

[11] P. Srivastava y N. Singh, «Automatized Medical Chatbot (Medibot)», en *2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC)*, feb. 2020, pp. 351-354. doi: 10.1109/PARC49193.2020.236624.

[12] R. B. Mathew, S. Varghese, S. E. Joy, y S. S. Alex, «Chatbot for Disease Prediction and Treatment Recommendation using Machine Learning», en *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, abr. 2019, pp. 851-856. doi: 10.1109/ICOEI.2019.8862707.

[13] «Large-scale epidemiological monitoring of the COVID-19 epidemic in Tokyo - The Lancet Regional Health – Western Pacific». Accedido: 21 de octubre de 2024. [En línea]. Disponible en: https://www.thelancet.com/journals/lanwpc/article/PIIS2666-6065(20)30016-X/fulltext

[14] L. P. Diaz Tito, J. V. Tito Cárdenas, G. Garcia Curo, y A. M. Boy Barreto, «Inteligencia artificial aplicada al sector educativo», *Rev. Venez. Gerenc. RVG*, vol. 26, n.o 96, pp. 1189-1200, 2021.

[15] C. G. Mont, C. M. D. Pozo, C. M. Pinto, y A. V. M. del C. Alcocer, «Artificial Intelligence for Social Good in Latin America and the Caribbean: The Regional Landscape and 12 Country Snapshots», *IDB Publ.*, may 2020, doi: 10.18235/0002393.

[16] J. E. G. Mateos, «El ABC de los ChatBots en la educación : una a proximación», *Rev. Cuba. Educ. Super.*, vol. 42, n.o especial 1, Art. n.o especial 1, oct. 2023.

[17] M. S. Rodríguez Parrales, «Desarrollo de una aplicación web para la gestión bibliotecaria de un Instituto Tecnológico aplicando un asistente virtual Chatbots con reconocimiento de voz.», bachelorThesis, La Libertad: Universidad Estatal Península de Santa Elena, 2024., 2024. Accedido: 21 de octubre de 2024. [En línea]. Disponible en: https://repositorio.upse.edu.ec/handle/46000/10930

[18] H. A. Martínez Torres, «Propuesta de Chatbot para mejorar los servicios digitales de la Biblioteca Octavio Arizmendi Posada de la Universidad de La Sabana», master thesis, Universidad de La Sabana, 2022. Accedido: 21 de octubre de 2024. [En línea]. Disponible en: https://intellectum.unisabana.edu.co/handle/10818/52557

[19] Yeray Expósito García, «Bullbot: asistente virtual para la gestión de servicios de la Biblioteca de la ULL.» Accedido: 21 de octubre de 2024. [En línea]. Disponible en: https://riull.ull.es/xmlui/handle/915/14747

[20] C. G. Hidalgo Suárez, J. M. Llanos Mosquera, y V. A. Bucheli Guerrero, «Una revisión sistemática sobre aula invertida y aprendizaje colaborativo apoyados en inteligencia artificial para el aprendizaje de programación», *Tecnura*, vol. 25, n.o 69, pp. 196-214, sep. 2021, doi: 10.14483/22487638.16934.

[21] J. M. Gutiérrez Siliceo, «Desarrollo de chatbots con entornos de código abierto», *Chatbots development with open source frameworks*, jul. 2019, Accedido: 10 de noviembre de 2024. [En línea]. Disponible en: https://repositorio.unican.es/xmlui/handle/10902/16456

[22] «Implementing Conversational AI in Libraries: A Practical Approach by Subhajit Panda, Prof. Rupak Chakravarty :: SSRN». Accedido: 10 de noviembre de 2024. [En línea]. Disponible en: https://papers.ssrn.com/sol3/papers.cfm?abstract\_id=3845429

[23] P. Guijarro Marco, «Asistente virtual para un sistema de información», ene. 2020, Accedido: 26 de octubre de 2024. [En línea]. Disponible en: http://rua.ua.es/dspace/handle/10045/101934

[24] J. M. Rubio *et al.*, «Proyecto UBOT: asistente virtual para entornos virtuales de aprendizaje», *Inf. Tecnológica*, vol. 33, n.o 4, pp. 85-92, ago. 2022, doi: 10.4067/S0718-07642022000400085.

[25] F. Colace, M. D. Santo, M. Lombardi, F. Pascale, A. Pietrosanto, y S. Lemma, «Chatbot for E-Learning: A Case of Study», *Int. J. Mech. Eng. Robot. Res.*, pp. 528-533, 2018, doi: 10.18178/ijmerr.7.5.528-533.

[26] B. Galitsky, «Chatbot Components and Architectures», en *Developing Enterprise Chatbots: Learning Linguistic Structures*, B. Galitsky, Ed., Cham: Springer International Publishing, 2019, pp. 13-51. doi: 10.1007/978-3-030-04299-8\_2.

[27] J. Huang, M. Zhou, y D. Yang, «Extracting chatbot knowledge from online discussion forums», en *Proceedings of the 20th international joint conference on Artifical intelligence*, en IJCAI’07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., ene. 2007, pp. 423-428.

[28] A. Abashev, R. Grigoryev, K. Grigorian, y V. Boyko, «Programming Tools for Messenger-Based Chatbot System Organization: Implication for Outpatient and Translational Medicines», *BioNanoScience*, vol. 2, n.o 7, pp. 403-407, nov. 2016, doi: 10.1007/s12668-016-0376-9.

[29] M. Dibitonto, K. Leszczynska, F. Tazzi, y C. M. Medaglia, «Chatbot in a Campus Environment: Design of LiSA, a Virtual Assistant to Help Students in Their University Life», en *Human-Computer Interaction. Interaction Technologies*, M. Kurosu, Ed., Cham: Springer International Publishing, 2018, pp. 103-116. doi: 10.1007/978-3-319-91250-9\_9.

[30] A. Følstad, M. Skjuve, y P. B. Brandtzaeg, «Different Chatbots for Different Purposes: Towards a Typology of Chatbots to Understand Interaction Design», en *Internet Science*, S. S. Bodrunova, O. Koltsova, A. Følstad, H. Halpin, P. Kolozaridi, L. Yuldashev, A. Smoliarova, y H. Niedermayer, Eds., Cham: Springer International Publishing, 2019, pp. 145-156. doi: 10.1007/978-3-030-17705-8\_13.

[31] K. Kuligowska, «Commercial Chatbot: Performance Evaluation, Usability Metrics and Quality Standards of Embodied Conversational Agents», 5 de febrero de 2015, *Social Science Research Network, Rochester, NY*: 2569637. Accedido: 10 de noviembre de 2024. [En línea]. Disponible en: https://papers.ssrn.com/abstract=2569637

[32] A. P. Chaves y M. A. Gerosa, «How Should My Chatbot Interact? A Survey on Social Characteristics in Human–Chatbot Interaction Design», *Int. J. Human–Computer Interact.*, vol. 37, n.o 8, pp. 729-758, may 2021, doi: 10.1080/10447318.2020.1841438.

[33] M. Jain, P. Kumar, R. Kota, y S. N. Patel, «Evaluating and Informing the Design of Chatbots», en *Proceedings of the 2018 Designing Interactive Systems Conference*, en DIS ’18. New York, NY, USA: Association for Computing Machinery, jun. 2018, pp. 895-906. doi: 10.1145/3196709.3196735.

[34] S. Kiousis, «Interactivity: a concept explication», *New Media Soc.*, vol. 4, n.o 3, pp. 355-383, sep. 2002, doi: 10.1177/146144480200400303.

[35] S. Diederich, A. Brendel, y L. Kolbe, «Towards a Taxonomy of Platforms for Conversational Agent Design», *Wirtsch. 2019 Proc.*, feb. 2019, [En línea]. Disponible en: https://aisel.aisnet.org/wi2019/track10/papers/1

[36] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, y P. Steggles, «Towards a Better Understanding of Context and Context-Awareness», en *Handheld and Ubiquitous Computing*, vol. 1707, H.-W. Gellersen, Ed., en Lecture Notes in Computer Science, vol. 1707. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 304-307. doi: 10.1007/3-540-48157-5\_29.

[37] E. Adamopoulou y L. Moussiades, «An Overview of Chatbot Technology», en *Artificial Intelligence Applications and Innovations*, I. Maglogiannis, L. Iliadis, y E. Pimenidis, Eds., Cham: Springer International Publishing, 2020, pp. 373-383. doi: 10.1007/978-3-030-49186-4\_31.

[38] P. Kucherbaev, A. Bozzon, y G.-J. Houben, «Human-Aided Bots», *IEEE Internet Comput.*, vol. 22, n.o 6, pp. 36-43, nov. 2018, doi: 10.1109/MIC.2018.252095348.

[39] K. Ramesh, S. Ravishankaran, A. Joshi, y K. Chandrasekaran, «A Survey of Design Techniques for Conversational Agents», en *Information, Communication and Computing Technology*, S. Kaushik, D. Gupta, L. Kharb, y D. Chahal, Eds., Singapore: Springer, 2017, pp. 336-350. doi: 10.1007/978-981-10-6544-6\_31.

[40] Y. Wu, W. Wu, C. Xing, M. Zhou, y Z. Li, «Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-based Chatbots», 14 de mayo de 2017, *arXiv*: arXiv:1612.01627. doi: 10.48550/arXiv.1612.01627.

[41] S. B. Dodda, S. Maruthi, R. R. Yellu, P. Thuniki, y S. R. B. Reddy, «Conversational AI - Chatbot Architectures and Evaluation: Analyzing architectures and evaluation methods for conversational AI systems, including chatbots, virtual assistants, and dialogue systems», *Aust. J. Mach. Learn. Res. Appl.*, vol. 1, n.o 1, Art. n.o 1, jun. 2021.

[42] H. T. Hien, P.-N. Cuong, L. N. H. Nam, H. L. T. K. Nhung, y L. D. Thang, «Intelligent Assistants in Higher-Education Environments: The FIT-EBot, a Chatbot for Administrative and Learning Support», en *Proceedings of the 9th International Symposium on Information and Communication Technology*, en SoICT ’18. New York, NY, USA: Association for Computing Machinery, dic. 2018, pp. 69-76. doi: 10.1145/3287921.3287937.

[43] «Data science in light of natural language processing: An overview», *Procedia Comput. Sci.*, vol. 127, pp. 82-91, ene. 2018, doi: 10.1016/j.procs.2018.01.101.

[44] L. Deng y Y. Liu, *Deep Learning in Natural Language Processing*. Springer, 2018.

[45] B. (Borja) Quevedo-Marcos, «Análisis de las herrramientas de procesamiento de lenguaje natural para estructurar textos médicos.», oct. 2020, Accedido: 11 de noviembre de 2024. [En línea]. Disponible en: https://dadun.unav.edu/handle/10171/60003

[46] H. A. Gómez-Loizaga, «Agente conversacional para búsqueda y recomendación de ofertas laborales», masterThesis, 2019. Accedido: 11 de noviembre de 2024. [En línea]. Disponible en: https://reunir.unir.net/handle/123456789/9564

[47] A. L. Samuel, «Some Studies in Machine Learning Using the Game of Checkers», *IBM J. Res. Dev.*, vol. 3, n.o 3, pp. 210-229, jul. 1959, doi: 10.1147/rd.33.0210.

[48] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. USA: Prentice Hall PTR, 1998.

[49] A. Vaswani *et al.*, «Attention is All you Need».

[50] S. Sánchez García y E. Valveny Llobet, «Integración de arquitectura de generación aumentada por recuperación (RAG) en la plataforma Wattwin», 2024.

[51] P. Lewis *et al.*, «Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks», en *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, pp. 9459-9474. Accedido: 12 de noviembre de 2024. [En línea]. Disponible en: https://proceedings.neurips.cc/paper\_files/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[52] «Introduction | 🦜️🔗 LangChain». Accedido: 11 de noviembre de 2024. [En línea]. Disponible en: https://python.langchain.com/docs/introduction/

[53] S. Goonatilleke y B. Hettige, «Past, Present and Future Trends in Multi-Agent System Technology», *J. Eur. Systèmes Autom.*, vol. 55, pp. 723-739, dic. 2022, doi: 10.18280/jesa.550604.

[54] R. C. Cardoso y A. Ferrando, «A Review of Agent-Based Programming for Multi-Agent Systems», *Computers*, vol. 10, n.o 2, Art. n.o 2, feb. 2021, doi: 10.3390/computers10020016.

[55] P. Maes, «Artificial life meets entertainment: lifelike autonomous agents», *Commun. ACM*, vol. 38, n.o 11, pp. 108-114, nov. 1995, doi: 10.1145/219717.219808.

[56] «Welcome to the Foundation for Intelligent Physical Agents». Accedido: 7 de febrero de 2024. [En línea]. Disponible en: http://www.fipa.org/

[57] M. Wooldridge y N. R. Jennings, «Intelligent agents: theory and practice», *Knowl. Eng. Rev.*, vol. 10, n.o 2, pp. 115-152, jun. 1995, doi: 10.1017/S0269888900008122.

[58] M. Joshi, «Automated Chatbots for Improved User Services in University Libraries», *Libr. Prog. Int.*, vol. 44, n.o 1, Art. n.o 1, jul. 2024.

[59] Y. A. Lee, «Investigating AI Chatbot Integration in Academic Libraries: A Case Study», ago. 2024, Accedido: 26 de octubre de 2024. [En línea]. Disponible en: https://shareok.org/handle/11244/340414

[60] S. Rodriguez y C. Mune, «Uncoding library chatbots: deploying a new virtual reference tool at the San Jose State University library», *Ref. Serv. Rev.*, vol. 50, n.o 3/4, pp. 392-405, oct. 2022, doi: 10.1108/RSR-05-2022-0020.

[61] T. L. Reinsfelder y K. O’Hara-Krebs, «Implementing a Rules-Based Chatbot for Reference Service at a Large University Library», *J. Web Librariansh.*, vol. 17, n.o 4, pp. 95-109, oct. 2023, doi: 10.1080/19322909.2023.2268832.

[62] M. M. E. Torres y R. Manjarrés-Betancur, «Asistente Virtual Académico Utilizando Tecnologías Cognitivas De Procesamiento De Lenguaje Natural», *Rev. Politécnica*, vol. 16, n.o 31, pp. 85-96, 2020.

[63] «Dialogflow Documentation», Google Cloud. Accedido: 11 de noviembre de 2024. [En línea]. Disponible en: https://cloud.google.com/dialogflow/docs?hl=es-419

[64] «IBM Watson». Accedido: 11 de noviembre de 2024. [En línea]. Disponible en: https://www.ibm.com/watson

[65] «Chatbot | Deep learning | Amazon Lex», Amazon Web Services, Inc. Accedido: 11 de noviembre de 2024. [En línea]. Disponible en: https://aws.amazon.com/es/lex/

[66] «Introduction to Rasa Open Source & Rasa Pro». Accedido: 11 de noviembre de 2024. [En línea]. Disponible en: https://rasa.com/docs/rasa/

[67] «Ollama». Accedido: 11 de noviembre de 2024. [En línea]. Disponible en: https://ollama.com

[68] «Nomic\_Ada\_Wiki | Atlas Map». Accedido: 7 de enero de 2025. [En línea]. Disponible en: https://atlas.nomic.ai/data/brandon/nomic-ada-wiki/map

[69] «Introducing Llama 3.1: Our most capable models to date», Meta AI. Accedido: 7 de enero de 2025. [En línea]. Disponible en: https://ai.meta.com/blog/meta-llama-3-1/

[70] «Chroma Docs». Accedido: 12 de noviembre de 2024. [En línea]. Disponible en: https://docs.trychroma.com/

[71] «The vector database to build knowledgeable AI | Pinecone». Accedido: 12 de noviembre de 2024. [En línea]. Disponible en: https://www.pinecone.io/

[72] «Open Source Vector Database | Weaviate». Accedido: 7 de enero de 2025. [En línea]. Disponible en: https://weaviate.io/platform?utm\_source=google&utm\_medium=cpc&utm\_campaign=Brand\_Weaviate\_Search&utm\_content=Weav\_VectorDB\_SKAG&utm\_term=weaviate%20vector%20database&utm\_campaign=Brand\_Weaviate\_Search&utm\_source=adwords&utm\_medium=ppc&hsa\_acc=3045935254&hsa\_cam=21749609685&hsa\_grp=169496070913&hsa\_ad=715211508191&hsa\_src=g&hsa\_tgt=kwd-2242793536970&hsa\_kw=weaviate%20vector%20database&hsa\_mt=e&hsa\_net=adwords&hsa\_ver=3&gad\_source=1&gclid=Cj0KCQiAvvO7BhC-ARIsAGFyToX60I0KforQlDahBzxv-cFIzzsMZZ5f5Ss\_6lklcxcjhcVpnSZHwPgaAu7QEALw\_wcB

[73] «Milvus | Open-source Vector Database created by Zilliz». Accedido: 7 de enero de 2025. [En línea]. Disponible en: https://zilliz.com/what-is-milvus?utm\_term=milvus%20vector%20database&utm\_campaign=WP\_Milvus\_Search&utm\_source=adwords&utm\_medium=ppc&hsa\_acc=3636806625&hsa\_cam=20033301076&hsa\_grp=153576449172&hsa\_ad=697853242445&hsa\_src=g&hsa\_tgt=kwd-1636564288935&hsa\_kw=milvus%20vector%20database&hsa\_mt=e&hsa\_net=adwords&hsa\_ver=3&gad\_source=1&gclid=Cj0KCQiAvvO7BhC-ARIsAGFyToWz5Sm\_1tTCBA3eDiKzDDCPoOc\_suDmTwsEMQPQi\_a3dxzVZoCIUQgaAnXMEALw\_wcB

[74] «FAISS». Accedido: 7 de enero de 2025. [En línea]. Disponible en: https://ai.meta.com/tools/faiss

[75] «MongoDB: La plataforma de datos para desarrolladores | MongoDB». Accedido: 12 de noviembre de 2024. [En línea]. Disponible en: https://www.mongodb.com/es

[76] «FastAPI». Accedido: 7 de enero de 2025. [En línea]. Disponible en: https://fastapi.tiangolo.com/

[77] L. C. Escalante, «El patrón de arquitectura n-capas con orientación al dominio como solución en el diseño de aplicaciones empresariales», *Tecnol. Desarro. Trujillo*, vol. 11, n.o 1, Art. n.o 1, dic. 2013, doi: 10.18050/td.v11i1.679.

[78] M. D. Hanson, «The Client/Server Architecture», en *Server Management*, Auerbach Publications, 2000.

[79] V. Jagannathan, *Blackboard Architectures and Applications*. Elsevier, 1989.

[80] R. S. Pressman, *Software engineering: a practitioner’s approach*, 7th ed. Dubuque, IA: McGraw-Hill, 2010.