PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
FACULTAD DE INGENIERÍA
CARRERA DE: SISTEMAS DE INFORMACIÓN

Trabajo de Titulación

DESARROLLO DE UN PROTOTIPO DE UN ASISTENTE DE INTELIGENCIA ARTIFICIAL PARA DOCUMENTACIÓN NORMATIVA DE LA PUCE.

AUTOR:
GABRIEL MATIAS GALLARDO CAMACHO

QUITO DM, ENERO DE 2026

Tutor: Edison Vicente Londoño Mora

Contenido

CAPITULO I: INTRODUCCIÓN 8
1.1 JUSTIFICACIÓN 8
1.2 PLANTEAMIENTO DEL PROBLEMA 9
1.3 OBJETIVOS 10
1.3.1 OBJETIVO GENERAL 10
1.3.2 OBJETIVOS ESPECIFICOS 10
1.4 ALCANCE 11
1.5 METODOLOGÍA DE DESARROLLO DEL SISTEMA 12
CAPITULO II: MARCO TEÓRICO Y CONCEPTUAL 12
2.1 ANTECEDENTES O MARCO REFERENCIAL 12
2.2 MARCO TEÓRICO 13
2.2.1 Inteligencia Artificial en la gestión de la información 13
2.2.2 Modelos de Lenguaje de Gran Escala (LLM) 13
2.2.3 Recuperación Aumentada por Generación (RAG) 13
2.2.4 Embeddings y búsqueda semántica 14
2.2.5 Asistentes de inteligencia artificial en entornos universitarios 15
2.2.6 Gestión documental institucional apoyada en inteligencia artificial 16
2.3 Marco Conceptual 16
2.4 Antecedentes 17
2.5 Marco metodológico 18
2.5.1 Selección de la metodología 18
2.5.2 Roles, artefactos y eventos 19
CAPITULO III: ANÁLISIS, DISEÑO Y DESARROLLO DEL PROTOTIPO 20
3.1 ANÁLISIS DE REQUERIMIENTOS 20
3.1.1 Requerimientos funcionales 20

3.1.2 Requerimientos no funcionales 21
3.2 ARQUITECTURA DEL SISTEMA 23
3.2.1 Arquitectura general del sistema 23
3.2.2 Arquitectura basada en Recuperación Aumentada por Generación (RAG) 25
3.3 MODELAMIENTO 27
3.3.1 Casos de Uso 27
3.4 DISEÑO DEL PIPELINE RAG 30
3.4.1 Pipeline de construcción de la base de conocimiento 30
3.4.2 Pipeline de consulta y generación de respuestas 32
3.5 DISEÑO DE LA BASE DE DATOS VECTORIAL 34
3.5.1 Propósito y alcance de la base vectorial 34
3.5.2 Unidad de almacenamiento (documento y fragmento) 35
3.5.3 Proceso de indexación vectorial 35
3.5.4 Consulta y recuperación (búsqueda top-k) 35
3.6 DISEÑO DE LA INTERFAZ DEL USUARIO (UI) 37
3.6.1 Principios de diseño de la interfaz 37
3.6.2 Componentes principales de la interfaz 38
3.6.3 Flujo de interacción del usuario 39
3.6.4 Alcance del diseño de la interfaz 39
CAPITULO IV: DESARROLLO E IMPLEMENTACIÓN DEL PROTOTIPO 39
4.1 Tecnologías y herramientas utilizadas 40
4.1.1 Lenguaje de programación 40
4.1.2 Editor de texto / Entorno de desarrollo integrado 40
4.1.3 Modelo de lenguaje 40
4.1.4 Modelo de embeddings 41
4.1.5 Base de datos vectorial 41
4.1.6 Procesamiento de documentos 41

4.1.7 Control de versiones 42
4.2 ORGANIZACIÓN DEL DESARROLLO POR ITERACIONES 42
4.2.1 Iteración 1: Preparación del entorno y base de conocimiento 42
4.2.2 Iteración 2: Implementación del pipeline RAG 43
4.2.3 Iteración 3: Integración, interfaz y validación 44
4.3 IMPLEMENTACIÓN DEL MÓDULO DE INGESTA DOCUMENTAL 46
4.3.1 Extracción de texto desde documentos PDF 47
4.3.2 Normalización y limpieza del texto normativo 47
4.3.3 Segmentación normativa (chunking) 48
4.3.4 Generación de embeddings de fragmentos normativos 48
4.3.5 Almacenamiento e indexación en FAISS 49
4.4 IMPLEMENTACIÓN DE LA RECUPERACIÓN SEMÁNTICA Y GENERACIÓN DE RESPUESTAS 50
4.4.1 Vectorización de la consulta del usuario 50
4.4.2 Recuperación de fragmentos normativos mediante FAISS 51
4.4.3 Construcción del contexto normativo 51
4.4.4 Generación de respuesta con ChatGPT mediante API 52
4.5 Implementación de la interfaz del usuario e integración final 53
4.5.1 Interfaz del usuario 54
CAPÍTULO V: PRUEBAS Y RESULTADOS 55
5.1 Plan de pruebas 55
5.2 Pruebas funcionales del sistema 56
5.3 Casos de prueba aplicados 59
5.4 Resultados obtenidos 61
5.5 Validación del cumplimiento de requerimientos 62
5.6 Discusión y limitaciones de la validación 62
CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES 63

6.1 Conclusiones 63
6.2 Recomendaciones.....64
REFERENCIAS.....65

Índice de Figuras

Figura 1 Concepto RAG 14
Figura 2 Concepto Embeddings y Búsqueda Semántica 15
Figura 3 Concepto Arquitectura Modular 23
Figura 4 Arquitectura General del Prototipo 25
Figura 5 Flujo de Funcionamiento 26
Figura 6 Casos de Uso 27
Figura 7 Transformación a Embedding 31
Figura 8 Almacenamiento en Base de Datos Vectorial 32
Figura 9 Diagrama Pipeline RAG 34
Figura 10 Diseño Base Vectorial 36
Figura 11 Ejemplo de UI 38
Figura 12 Preparación de Entorno 43
Figura 13 Fragmento de Código - Implementación del pipeline RAG 44
Figura 14 Primer Prototipo y UI para pruebas 45
Figura 15 Segundo Prototipo y UI para pruebas 45
Figura 16 Tercer Prototipo y UI para pruebas 46
Figura 17 Fragmento de Código - Extracción de texto desde documentos PDF 47
Figura 18 Fragmento de Código - Normalización y limpieza del texto normativo 47
Figura 19 Fragmento de Código - Segmentación normativa (chunking) 48
Figura 20 Fragmento de Código - Generación de embeddings de fragmentos normativos 49
Figura 21 Fragmento de Código - Almacenamiento e indexación en FAISS 49
Figura 22 Fragmento de Código - Vectorización de la consulta del usuario 50
Figura 23 Fragmento de Código - Recuperación de fragmentos normativos mediante FAISS 51
Figura 24 Fragmento de Código - Construcción del contexto normativo 52
Figura 25 Fragmento de Código - Generación de respuesta con ChatGPT mediante API 53
Figura 26 Fragmento de código UI 54
Figura 27 Prueba 1 – Funcionamiento del Sistema 56
Figura 28 Prueba 2 – Funcionamiento del Sistema 58
Figura 29 Prueba 3 – Funcionamiento del Sistema 58

Índice de Tablas

Tabla 1 Requerimientos Funcionales 21
Tabla 2 Requerimientos no funcionales 22
Tabla 3 Caso de Uso UC-01: Realizar consulta normativa 28
Tabla 4 Caso de Uso UC-02: Visualizar fuentes normativas 29
Tabla 5 Caso de Uso UC-03: Cargar documentos normativos 29
Tabla 6 Prueba 1 – Resultado 57
Tabla 7 Prueba 2 – Resultado 58
Tabla 8 Prueba 3 – Resultado 59
Tabla 9 Comparativa Resultado Esperado – Resultado Obtenido 61
Tabla 10 Validación cumplimiento de requerimientos 62

CAPITULO I: INTRODUCCIÓN

DESARROLLO DE UN PROTOTIPO DE UN ASISTENTE DE INTELIGENCIA ARTIFICIAL PARA DOCUMENTACIÓN NORMATIVA DE LA PUCE.

1.1 JUSTIFICACIÓN

En la actualidad las instituciones de educación superior gestionan un volumen creciente de documentación que abarca normativas, reglamentos, manuales, políticas internas, procedimientos académicos y administrativos.

En el caso de la Pontificia Universidad Católica del Ecuador (PUCE) la información se encuentra distribuida en múltiples repositorios, básicamente descentralizada, lo que dificulta su acceso rápido, preciso y centralizado tanto para estudiantes como para docentes y personal administrativo, esta dispersión documental puede generar demoras en los procesos y una menor eficiencia en la atención de consultas institucionales.

Con el avance de la inteligencia artificial y el procesamiento del lenguaje natural, se abren nuevas posibilidades para optimizar la gestión y consulta de información institucional, particularmente hablando del uso de modelos de lenguaje de gran escala (LLMs) combinados con técnicas de Generación Aumentada por Recuperación (RAG) permite crear asistentes virtuales capaces de comprender preguntas en lenguaje natural y ofrecer respuestas precisas basadas en la documentación oficial, sin necesidad de búsqueda manual.

Este proyecto de investigación plantea el desarrollo de un prototipo de asistente de inteligencia artificial especializado en la documentación normativa institucional de la PUCE, el mismo que se implementará mediante el consumo de la API de ChatGPT y el uso de embeddings para el procesamiento semántico de los textos.

A través de esta solución, se busca brindar una herramienta segura, confiable y eficiente que facilite la interacción con la información universitaria, promoviendo la efectividad, la transparencia y la reducción de tiempos de búsqueda.

Como resultado, se espera obtener un prototipo de sistema experto que no solo simplifique el acceso a los documentos oficiales, sino que también fortalezca la transformación digital de la PUCE. Sirviendo así como base para futuros desarrollos de inteligencia artificial institucional orientados a la automatización, la innovación y la mejora continua en la gestión del conocimiento universitario.

1.2 PLANTEAMIENTO DEL PROBLEMA

En el entorno universitario actual, la gestión documental representa un desafío constante para las instituciones de educación superior. La Pontificia Universidad Católica del Ecuador (PUCE) dispone de una amplia cantidad de documentos institucionales distribuidos en diferentes plataformas, formatos y repositorios. Esta misma dispersión dificulta la localización rápida de información específica y genera inconvenientes tanto para los estudiantes como para el personal docente y/o administrativo, quienes tal vez no diariamente pero ocasionalmente necesitan acceder a esta información de manera confiable y actualizada para realizar sus actividades profesionales.

A pesar de los avances tecnológicos y la disponibilidad de sistemas de gestión documental, la búsqueda y recuperación de información en la PUCE continúa siendo, en muchos casos, un proceso manual y poco eficiente. Los usuarios deben navegar entre múltiples fuentes, documentos extensos o sitios web sin un motor de búsqueda especifico capaz de interpretar sus consultas o inquietudes. Esta situación puede provocar pérdida de tiempo, errores de interpretación y una disminución en la productividad institucional general, tanto del lado del estudiante como del profesor o directivo. Además, la ausencia de un sistema inteligente que centralice y procese la documentación limita la capacidad de la universidad para ofrecer una atención ágil y automatizada a las consultas internas y externas.

Ante este contexto, surge la necesidad de desarrollar una solución tecnológica basada en inteligencia artificial que permita optimizar la interacción con la documentación institucional, de esta manera garantizando precisión, rapidez y seguridad en la obtención de la información institucional. La implementación de un asistente virtual especializado, apoyado en modelos de

lenguaje y en el proceso de Generación Aumentada por Recuperación (RAG), permitiría brindar respuestas contextualizadas y confiables a partir de los documentos oficiales de la universidad, reduciendo la carga de búsqueda manual y fortaleciendo la gestión del conocimiento.

A partir de esta problemática, se plantea la siguiente pregunta principal del proyecto:

- ¿Cómo puede un asistente de inteligencia artificial, basado en técnicas de recuperación aumentada (RAG), optimizar el acceso y la gestión de la documentación institucional en la PUCE?

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Desarrollar un prototipo de asistente de inteligencia artificial para la Pontificia Universidad Católica del Ecuador (PUCE) basado en técnicas de Generación Aumentada por Recuperación (RAG) y consumo de la API de ChatGPT, que facilite el acceso, comprensión y gestión de la documentación institucional de manera eficiente, segura y contextualizada.

1.3.2 OBJETIVOS ESPECIFICOS

- Identificar y clasificar las fuentes documentales relevantes que conformarán la base de conocimiento del asistente, garantizando la veracidad y vigencia de la información.
- Diseñar la arquitectura técnica del sistema, incorporando el modelo de recuperación aumentada (RAG), embeddings semánticos y esquemas de intercambio de datos en formato JSON.
- Desarrollar el prototipo del asistente mediante la integración de la API de ChatGPT, implementando mecanismos de búsqueda contextual y generación de respuestas basadas en documentos oficiales.

- Evaluar la precisión, eficiencia y utilidad del asistente mediante pruebas de funcionamiento que permitan validar su desempeño.

1.4 ALCANCE

El presente proyecto tiene como alcance el diseño y desarrollo de un prototipo funcional de un asistente de inteligencia artificial orientado exclusivamente a la consulta y recuperación de normativa institucional de la Pontificia Universidad Católica del Ecuador (PUCE), utilizando técnicas de Generación Aumentada por Recuperación (RAG) y el consumo de la API de ChatGPT.

El prototipo permitirá a los usuarios realizar consultas en lenguaje natural relacionadas con la normativa institucional de la PUCE, recuperando información relevante desde una base de conocimiento construida a partir de documentos oficiales normativos. Las respuestas generadas estarán fundamentadas en los textos normativos cargados en el sistema, garantizando coherencia, precisión y alineación con la normativa vigente.

El desarrollo del proyecto contempla las siguientes actividades principales:

- Identificación y selección de normativa institucional pública de la PUCE.
- Análisis y estructuración de documentos normativos para su procesamiento.
- Segmentación del contenido normativo y generación de embeddings semánticos.
- Implementación de un repositorio vectorial para la recuperación de información normativa.
- Integración de la API de ChatGPT para la generación de respuestas basadas en normativa institucional.
- Desarrollo de una interfaz básica de interacción para consultas normativas.
- Validación del prototipo mediante pruebas de consulta sobre normativa institucional.

1.5 METODOLOGÍA DE DESARROLLO DEL SISTEMA

Para el desarrollo del prototipo se adoptó una metodología Iterativa–Incremental con enfoque en prototipado, la misma que permite construir el sistema de manera progresiva mediante ciclos de desarrollo, prueba y mejora continua. Esta metodología resulta adecuada para proyectos de inteligencia artificial ya que facilita la experimentación, el ajuste de parámetros y la validación temprana de resultados.

Cada iteración del proceso incluyó actividades de análisis, diseño, implementación y evaluación, permitiendo refinar el asistente de inteligencia artificial en función de los resultados obtenidos y las necesidades identificadas. Este enfoque permitió reducir riesgos técnicos, mejorar la calidad de las respuestas generadas y asegurar la alineación del prototipo con los objetivos del proyecto.

CAPITULO II: MARCO TEÓRICO Y CONCEPTUAL

2.1 ANTECEDENTES O MARCO REFERENCIAL

“En nuestro negocio, hablamos sobre tecnologías emergentes y cómo impactan a la sociedad. Nunca hemos visto una tecnología que se mueva tan rápido como la IA tiene un impacto en la sociedad y la tecnología. Esta es, con mucho, la tecnología de movimiento más rápido que hemos rastreado en términos de su impacto y recién estamos comenzando”.

Human + machine: reimagining work in the age of AI | WorldCat.org. (2018).

https://search.worldcat.org/es/title/human-machine-reimagining-work-in-the-age-ofai/oclc/1023487947

2.2 MARCO TEÓRICO

2.2.1 Inteligencia Artificial en la gestión de la información

La inteligencia artificial (IA) se ha consolidado como una herramienta clave para la automatización y optimización de procesos relacionados con la gestión de información. Su aplicación permite analizar grandes volúmenes de datos, identificar patrones y ofrecer respuestas contextualizadas en tiempos reducidos. En el ámbito documental, la IA facilita la clasificación, recuperación y análisis de textos extensos, superando las limitaciones de los sistemas de búsqueda tradicionales basados únicamente en coincidencias literales.

En organizaciones complejas, como las instituciones de educación superior, la IA contribuye a mejorar la eficiencia operativa y la accesibilidad a la información normativa, permitiendo que los usuarios realicen consultas en lenguaje natural y obtengan respuestas claras y precisas.

2.2.2 Modelos de Lenguaje de Gran Escala (LLM)

Los modelos de lenguaje de gran escala (Large Language Models, LLM) son sistemas de inteligencia artificial entrenados con grandes volúmenes de texto, capaces de comprender, generar y razonar en lenguaje natural. Estos modelos aprenden patrones lingüísticos y semánticos que les permiten interpretar preguntas complejas y producir respuestas coherentes. Según Myers (2024), los LLM constituyen la base de herramientas como ChatGPT, las cuales pueden adaptarse a contextos específicos cuando se integran con fuentes de información controladas. En proyectos institucionales, los LLM permiten desarrollar asistentes especializados que responden de manera contextualizada y alineada con documentación oficial.

2.2.3 Recuperación Aumentada por Generación (RAG)

La Recuperación Aumentada por Generación (Retrieval-Augmented Generation, RAG) es un enfoque que combina la capacidad generativa de los modelos de lenguaje con mecanismos de

recuperación de información desde bases de conocimiento externas. Según Lewis et al. (2020), este paradigma permite generar respuestas basadas en documentos específicos, mejorando la precisión y confiabilidad de la información producida.

Merritt (2025) señala que RAG se ha convertido en un estándar dentro del desarrollo de sistemas de inteligencia artificial modernos, ya que reduce significativamente el riesgo de respuestas incorrectas o no verificadas. En el contexto institucional, esta técnica resulta especialmente útil para garantizar que las respuestas estén alineadas con normativas oficiales y no dependan exclusivamente del conocimiento general del modelo.

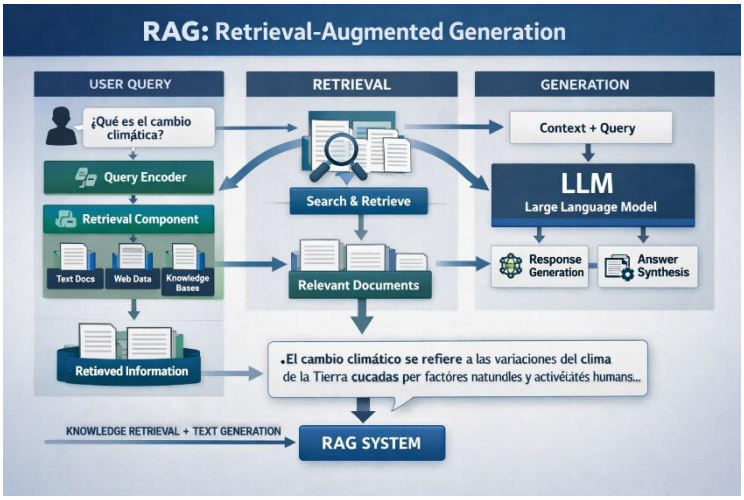

Figura 1 *Concepto RAG*

Fuente: Elaboración propia a través de DALL-E 3

2.2.4 Embeddings y búsqueda semántica

Los embeddings son representaciones vectoriales de texto que capturan su significado semántico, permitiendo comparar documentos y consultas mediante medidas de similitud. De acuerdo con Pinecone (s. f.), los vector embeddings convierten información textual en vectores de alta dimensión, facilitando operaciones como la búsqueda semántica y la clasificación de documentos.

Esta tecnología permite que un sistema identifique fragmentos relevantes incluso cuando las consultas no coinciden literalmente con el texto original, lo cual resulta fundamental para la consulta eficiente de documentos normativos extensos y complejos.

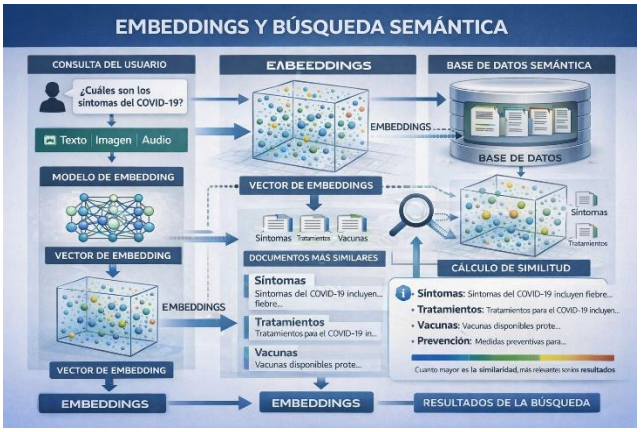

Figura 2 *Concepto Embeddings y Búsqueda Semántica*

Fuente: Elaboración propia a través de DALL-E 3

2.2.5 Asistentes de inteligencia artificial en entornos universitarios

Los asistentes de inteligencia artificial son aplicaciones diseñadas para interactuar con los usuarios mediante lenguaje natural, proporcionando respuestas contextualizadas y oportunas. En entornos universitarios, estos asistentes pueden apoyar procesos académicos y administrativos, desde la consulta de artículos y declaraciones hasta la simple solvencia de dudas, así facilitando el acceso a información institucional y reduciendo la carga operativa del personal.

Basado en lo expuesto por Merritt (2025), la integración de asistentes de IA con bases documentales controladas permite ofrecer información confiable y segura, especialmente cuando se trata de normativas y políticas institucionales.

2.2.6 Gestión documental institucional apoyada en inteligencia artificial

La gestión documental institucional comprende el conjunto de procesos destinados a la organización, almacenamiento, acceso y control de documentos oficiales de la universidad. En universidades, valga la redundancia, esta gestión resulta crítica debido a la cantidad y complejidad de la normativa vigente.

La aplicación de técnicas de inteligencia artificial, particularmente mediante asistentes basados en RAG y búsqueda semántica, permite optimizar estos procesos, mejorando la accesibilidad, trazabilidad y comprensión de la normativa institucional. Esto contribuye a una gestión del conocimiento más eficiente y alineada con las necesidades de la comunidad universitaria.

2.3 Marco Conceptual

- **Recuperación Aumentada por Generación (RAG)**
  Técnica que combina un sistema de recuperación de información con un modelo generativo de lenguaje, permitiendo generar respuestas fundamentadas en documentos específicos, incrementando la precisión y confiabilidad del sistema y de la información.

- **Embeddings de Texto**
  Representaciones vectoriales de fragmentos de texto que capturan su significado semántico y permiten medir la similitud entre documentos y consultas.

- **Modelos de Lenguaje de Gran Escala (LLM)**
  Sistemas de inteligencia artificial entrenados con grandes volúmenes de texto, capaces de comprender y generar lenguaje natural de forma coherente.

- **Asistente de Inteligencia Artificial**
  Aplicación que utiliza modelos de lenguaje y técnicas de recuperación de información para interactuar con los usuarios y responder consultas de manera contextualizada y eficiente.

- **Gestión Documental Institucional**
  Conjunto de procesos, normas y herramientas orientadas a la administración, organización y acceso eficiente a los documentos oficiales de una institución.

2.4 Antecedentes

En los últimos años, la inteligencia artificial ha adquirido un papel relevante en la gestión y recuperación de información, especialmente en contextos que manejan grandes volúmenes de documentación. El desarrollo de modelos de lenguaje de gran escala integrados con mecanismos de recuperación de información ha permitido mejorar la precisión y confiabilidad de los sistemas de consulta documental.

Lewis et al. (2020) proponen el enfoque de Recuperación Aumentada por Generación (RAG) como una solución para tareas intensivas en conocimiento, demostrando que la combinación de modelos generativos con bases documentales externas reduce la generación de información incorrecta y mejora la calidad de las respuestas. Este enfoque ha sido adoptado progresivamente en sistemas orientados a la consulta de documentación técnica y normativa.

Asimismo, Merritt (2025) destaca que las arquitecturas RAG permiten a las organizaciones utilizar modelos avanzados de lenguaje manteniendo el control sobre las fuentes de información, aspecto fundamental en entornos institucionales. Por su parte, el uso de embeddings de texto ha facilitado la búsqueda semántica eficiente de documentos extensos, superando las limitaciones de los sistemas tradicionales basados en palabras clave (Pinecone, s. f.).

Estos antecedentes evidencian que la implementación de un asistente de inteligencia artificial especializado en normativa institucional constituye una solución alineada con las tendencias actuales de investigación y desarrollo tecnológico, y resulta pertinente para instituciones y entes de educación superior como lo es la PUCE.

2.5 Marco metodológico

El marco metodológico establece el enfoque y los lineamientos que guían el desarrollo del presente proyecto, el cual se orienta a la construcción de un prototipo funcional de asistente de inteligencia artificial enfocado en la consulta de normativa institucional de la Pontificia Universidad Católica del Ecuador (PUCE).

Dada la naturaleza tecnológica, experimental y aplicada del proyecto, se opta por una metodología flexible que permita iterar sobre el diseño, implementación y evaluación del sistema, así asegurando su viabilidad dentro de las limitaciones de tiempo y alcance establecidas.

2.5.1 Selección de la metodología

Para el desarrollo del proyecto se seleccionó una metodología iterativa–incremental con enfoque en prototipado, la cual permite construir el sistema de forma progresiva mediante ciclos sucesivos de análisis, diseño, implementación y validación.

Esta metodología resulta adecuada para proyectos de inteligencia artificial debido a que:

- Permite realizar ajustes continuos en función de los resultados obtenidos.
- Facilita la experimentación con diferentes configuraciones del modelo y del proceso RAG.
- Reduce riesgos técnicos al validar tempranamente el funcionamiento del prototipo.
- Se adapta a proyectos académicos con alcance limitado y tiempos definidos.

Cada iteración del desarrollo permite incorporar mejoras en la recuperación de información y en la generación de respuestas. Asimismo también permite incorporar mejoras en la interacción con el usuario y en el sistema, garantizando la alineación del prototipo con los objetivos planteados previamente en este trabajo de titulación.

2.5.2 Roles, artefactos y eventos

Roles

Dado que el proyecto se desarrolla en un contexto académico individual, se definen los siguientes roles:

- **Investigador–desarrollador:** responsable del análisis, diseño, implementación y documentación del prototipo de asistente de inteligencia artificial.
- **Tutor académico:** encargado de orientar, revisar y validar el avance metodológico y técnico del proyecto.

Artefactos

Durante el desarrollo del proyecto se generan los siguientes artefactos:

- Documento de requisitos del sistema.
- Base documental normativa de la PUCE.
- Repositorio de embeddings y base vectorial.
- Prototipo funcional del asistente de inteligencia artificial.
- Código fuente del sistema.

Eventos

Los principales eventos considerados dentro de la metodología son:

- Revisión periódica del avance con el tutor académico.
- Iteraciones de desarrollo y prueba del prototipo.
- Aplicación de pruebas de validación funcional.
- Evaluación final del sistema desarrollado.

CAPITULO III: ANÁLISIS, DISEÑO Y DESARROLLO DEL PROTOTIPO

El presente capítulo aborda el análisis, diseño y desarrollo del prototipo de asistente de inteligencia artificial para la documentación normativa de la PUCE propuesto previamente en este mismo trabajo de titulación. Orientado, valga la redundancia, a la consulta de normativa institucional de la Pontificia Universidad Católica del Ecuador (PUCE). En esta sección se describen los requerimientos del sistema, la arquitectura y los componentes técnicos que conforman la solución, así como el proceso de implementación del prototipo basado en técnicas de Recuperación Aumentada por Generación (RAG), embeddings de texto y modelos de lenguaje, evidenciando la viabilidad técnica del sistema desarrollado.

3.1 ANÁLISIS DE REQUERIMIENTOS

3.1.1 Requerimientos funcionales

Los requerimientos funcionales describen las acciones y servicios que el sistema debe proporcionar al usuario y a los componentes internos del prototipo.

Código	Requerimiento	Descripción
RF-01	Carga de normativa institucional	El sistema debe permitir la carga y almacenamiento de documentos normativos institucionales públicos de la PUCE en formato digital.
RF-02	Procesamiento de documentos normativos	El sistema debe procesar los documentos cargados, segmentándolos en fragmentos adecuados para su análisis semántico.

RF-03	Generación de embeddings	El sistema debe generar representaciones vectoriales (embeddings) a partir de los fragmentos de normativa institucional.
RF-04	Almacenamiento en base vectorial	El sistema debe almacenar los embeddings generados en una base de datos vectorial para facilitar la recuperación de información.
RF-05	Consulta en lenguaje natural	El sistema debe permitir a los usuarios realizar consultas en lenguaje natural relacionadas con la normativa institucional de la PUCE.
RF-06	Recuperación semántica de información	El sistema debe recuperar los fragmentos normativos más relevantes utilizando técnicas de búsqueda semántica basadas en embeddings.
RF-07	Generación de respuestas basadas en normativa	El sistema debe generar respuestas fundamentadas exclusivamente en la información recuperada desde los documentos normativos.
RF-08	Presentación clara de respuestas	El sistema debe presentar las respuestas de forma clara, comprensible y accesible para usuarios sin conocimientos técnicos.
RF-09	Funcionamiento como prototipo académico	El sistema debe operar como un prototipo académico que demuestre la viabilidad del uso de inteligencia artificial para la consulta normativa.
RF-10	Validación funcional del sistema	El sistema debe permitir la realización de pruebas funcionales mediante consultas representativas sobre normativa institucional.

Tabla 1 Requerimientos Funcionales

3.1.2 Requerimientos no funcionales

Los requerimientos no funcionales establecen las condiciones de calidad, restricciones y características técnicas bajo las cuales debe operar el sistema.

Código	Requerimiento	Descripción
RNF-01	Uso exclusivo de normativa pública	El sistema debe operar únicamente con normativa institucional pública de la PUCE.
RNF-02	Protección de la información	El sistema no debe acceder, almacenar ni procesar información personal, sensible o confidencial.
RNF-03	Tiempo de respuesta adecuado	El sistema debe ofrecer tiempos de respuesta aceptables para consultas normativas realizadas por los usuarios.
RNF-04	Modularidad del sistema	El sistema debe ser diseñado de forma modular para facilitar su mantenimiento y posibles ampliaciones.
RNF-05	Consistencia de las respuestas	El sistema debe garantizar coherencia entre la consulta del usuario, la normativa recuperada y la respuesta generada.
RNF-06	Uso de tecnologías actuales	El sistema debe ser desarrollado utilizando tecnologías actuales de inteligencia artificial y procesamiento de lenguaje natural.
RNF-07	Funcionamiento en entorno académico	El sistema debe ejecutarse correctamente en un entorno académico de desarrollo.
RNF-08	Escalabilidad documental	El sistema debe permitir la incorporación futura de nuevos documentos normativos sin afectar su funcionamiento.
RNF-09	Usabilidad para estudiantes	El sistema debe ser fácil de usar y comprensible para estudiantes universitarios.
RNF-10	Consistencia terminológica	El sistema debe mantener uniformidad terminológica y documental en todas las respuestas generadas.

Tabla 2 Requerimientos no funcionales

3.2 ARQUITECTURA DEL SISTEMA

La arquitectura del sistema define la estructura general del prototipo de asistente de inteligencia artificial y la forma en que interactúan sus componentes para cumplir con los requerimientos establecidos. Esta arquitectura ha sido diseñada considerando el alcance del proyecto.

El sistema adopta una arquitectura modular lo que permite separar responsabilidades, facilitar el mantenimiento y posibilitar futuras ampliaciones. Cada módulo cumple una función específica dentro del flujo de procesamiento de la información, desde la ingesta de documentos normativos hasta la generación de respuestas al usuario.

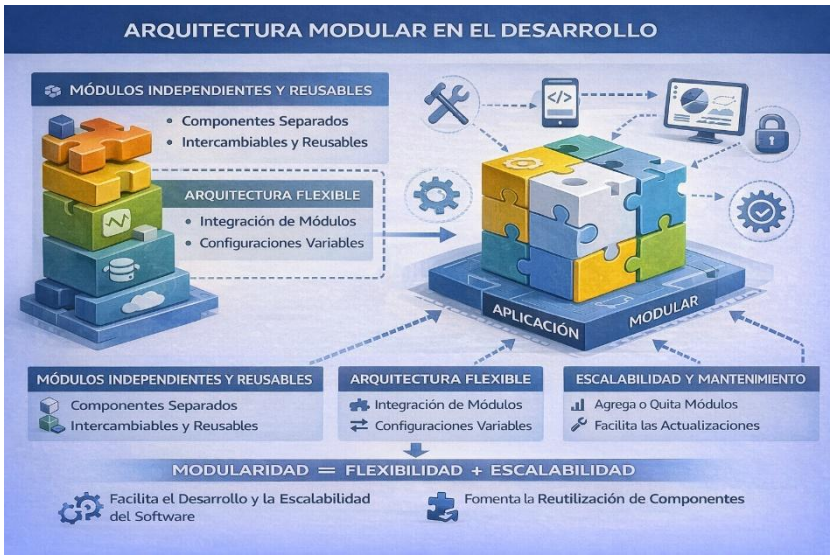

Figura 3 *Concepto Arquitectura Modular*

Fuente: Elaboración propia a través de DALL-E 3

3.2.1 Arquitectura general del sistema

De manera general, el prototipo está compuesto por los siguientes módulos principales:

1. **Módulo de Ingesta Documental**

   Encargado de la carga y gestión de los documentos normativos institucionales de la PUCE. Este módulo permite incorporar la normativa en formato digital y prepararla para su procesamiento posterior.

2. **Módulo de Procesamiento y Segmentación**

   Responsable de analizar los documentos normativos y dividirlos en fragmentos o secciones manejables, garantizando que cada segmento contenga información coherente y relevante para el análisis semántico.

3. **Módulo de Generación de Embeddings**

   Transforma los fragmentos de normativa en representaciones vectoriales mediante modelos de embeddings de texto, permitiendo capturar el significado semántico del contenido.

4. **Base de Datos Vectorial**

   Almacena los embeddings generados junto con la información asociada a cada fragmento normativo, posibilitando la búsqueda semántica eficiente.

5. **Módulo de Recuperación de Información**

   Recibe las consultas del usuario y recupera los fragmentos normativos más relevantes desde la base de datos vectorial, utilizando métricas de similitud.

6. **Módulo de Generación de Respuestas (LLM)**

   Integra el modelo de lenguaje a través de la API de OpenAI, utilizando los fragmentos recuperados como contexto para generar respuestas fundamentadas exclusivamente en la normativa institucional.

7. **Interfaz de Usuario**

   Permite la interacción entre el usuario y el sistema, facilitando el ingreso de consultas en lenguaje natural y la visualización de las respuestas generadas.

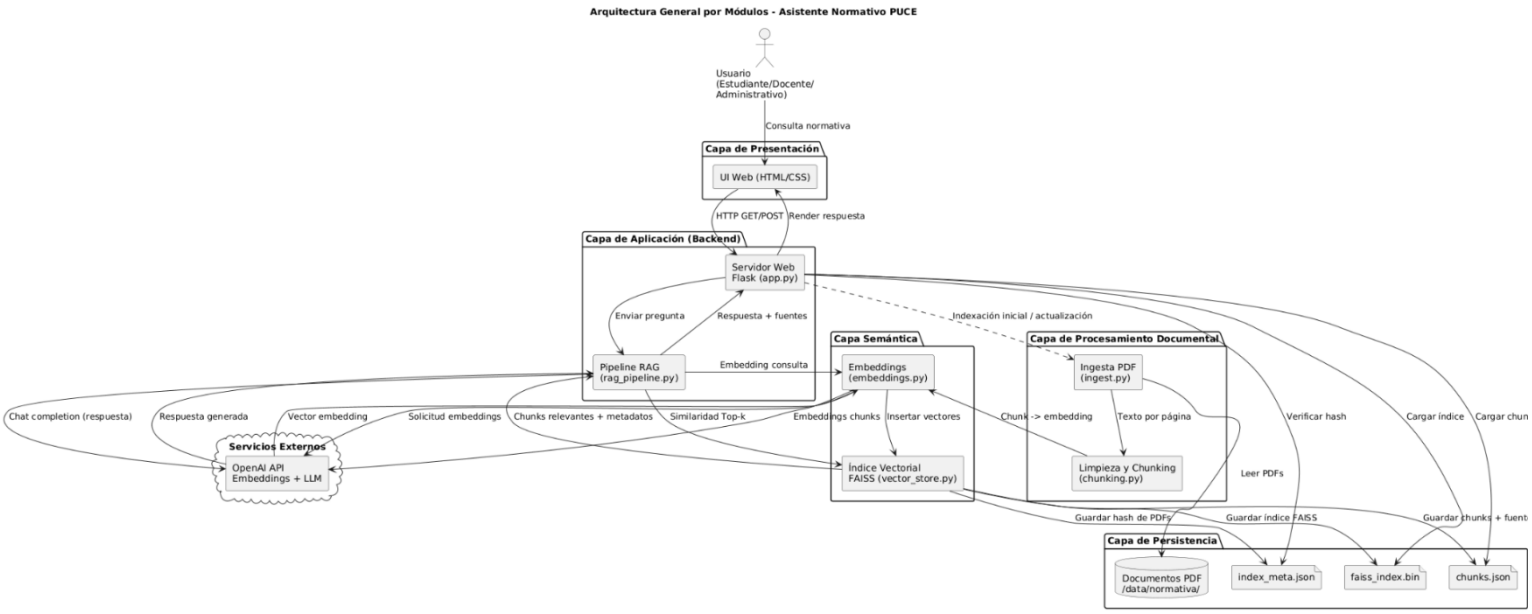

Figura 4 *Arquitectura General del Prototipo*

Fuente: Elaboración propia a través de PlantUML

3.2.2 Arquitectura basada en Recuperación Aumentada por Generación (RAG)

La arquitectura del sistema se fundamenta en el enfoque de Recuperación Aumentada por Generación (RAG), el cual combina un proceso de recuperación de información y un modelo generativo de lenguaje. Este enfoque garantiza que las respuestas proporcionadas por el asistente se basen en documentos normativos específicos y no únicamente en el conocimiento general del modelo.

El flujo de funcionamiento de la arquitectura RAG se desarrolla de la siguiente manera:

1. El usuario ingresa una consulta en lenguaje natural a través de la interfaz del sistema.

2. La consulta es transformada en un embedding utilizando el mismo modelo empleado para la normativa.

3. El sistema compara el embedding de la consulta con los embeddings almacenados en la base de datos vectorial.

4. Se recuperan los fragmentos normativos con mayor similitud semántica.

5. Los fragmentos recuperados se envían como contexto al modelo de lenguaje.

6. El modelo de lenguaje genera una respuesta coherente y contextualizada basada únicamente en la normativa recuperada.

7. La respuesta es presentada al usuario a través de la interfaz.

Este enfoque permite reducir la generación de respuestas incorrectas o no verificadas, asegurando que la información proporcionada sea consistente con la normativa institucional de la PUCE.

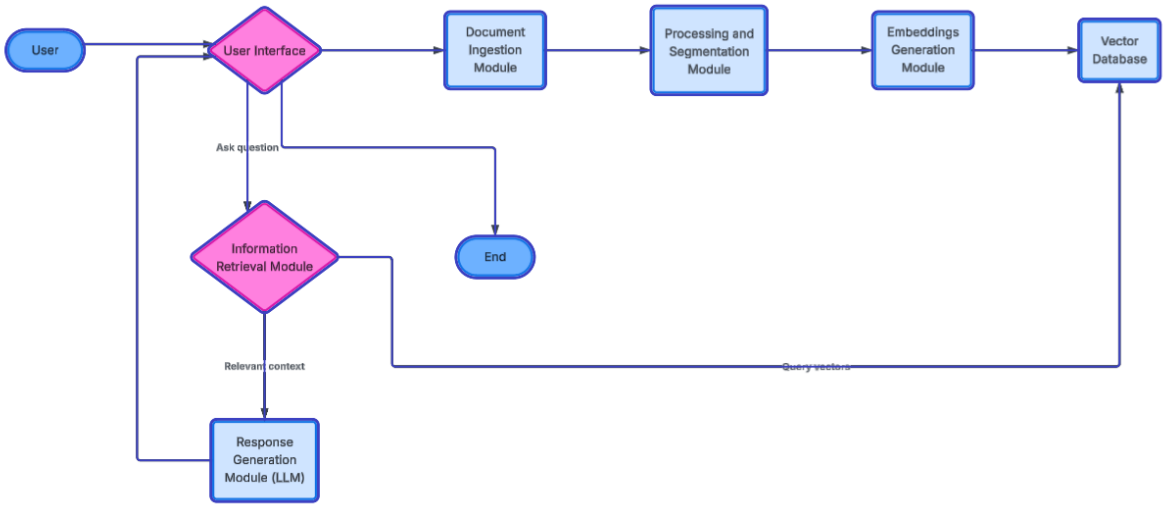

Figura 5 *Flujo de Funcionamiento*

Fuente: Elaboración propia a través de LucidChart.com

3.3 MODELAMIENTO

3.3.1 Casos de Uso

El diagrama de casos de uso representa la interacción funcional entre los actores y el sistema, en este caso se han identificado dos actores principales en la interacción con el sistema, el administrador y el estudiante/docente/administrativo.

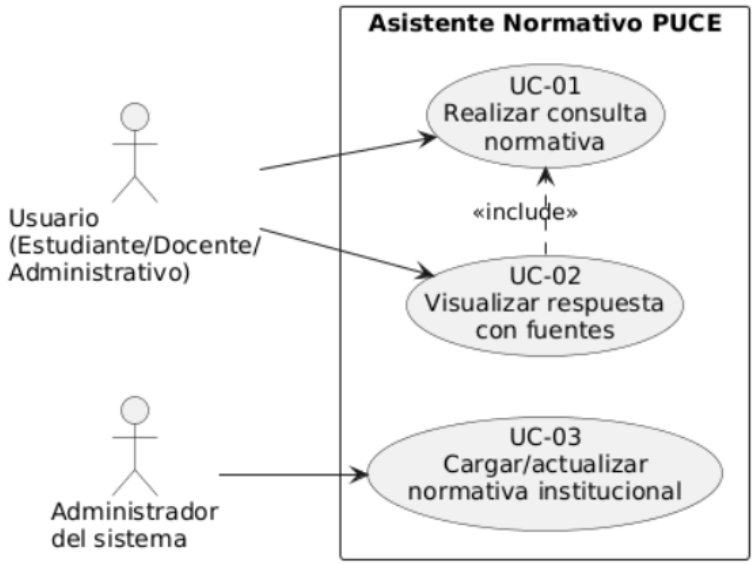

Figura 6 *Casos de Uso*

Fuente: Elaboración propia a través de PlantUML

Actores:

- Usuario (Estudiante/Docente/Administrativo): realiza consultas sobre normativa institucional.
- Administrador del sistema: gestiona la documentación normativa que alimenta al asistente.

Casos de uso:

1. Realizar consulta normativa: el usuario ingresa una pregunta en la interfaz y solicita una respuesta.

2. Visualizar respuesta: el sistema muestra la respuesta generada

3. Cargar/actualizar normativa institucional: el administrador incorpora nuevos PDFs normativos o reemplaza versiones y actualiza el índice vectorial.

Caso de Uso UC-01: Realizar consulta normativa

Elemento	**Descripción**
Código	UC-01
Nombre	Realizar consulta normativa
Actor principal	Usuario (Estudiante/Docente/Administrativo)
Descripción	Permite al usuario ingresar una pregunta relacionada con la normativa institucional de la PUCE para obtener una respuesta generada por el asistente.
Flujo principal	1. El usuario accede a la interfaz web. 2. Escribe una pregunta normativa. 3. Presiona el botón Consultar. 4. El sistema procesa la consulta y recupera fragmentos relevantes. 5. El sistema genera una respuesta basada en la normativa.

Tabla 3 Caso de Uso UC-01: Realizar consulta normativa

Caso de Uso UC-02: Visualizar fuentes normativas

Elemento	**Descripción**
Código	UC-02

Nombre	Visualizar fuentes normativas
Actor principal	Usuario (Estudiante/Docente/Administrativo)
Descripción	Permite al usuario los documentos normativos y páginas utilizadas como evidencia para sustentar la respuesta.
Flujo principal	1. El sistema entrega la respuesta generada. 2. El sistema muestra la lista de documentos normativos utilizados. 3. El usuario revisa las referencias y fuentes asociadas.

Tabla 4 Caso de Uso UC-02: Visualizar fuentes normativas

Caso de Uso UC-03: Cargar documentos normativos

Elemento	**Descripción**
Código	UC-03
Nombre	Cargar documentos normativos institucionales
Actor principal	Administrador del sistema
Descripción	Permite al administrador incorporar nuevos documentos normativos en formato PDF para que el asistente pueda utilizarlos como base de conocimiento.
Flujo principal	1. El administrador recopila documentos institucionales oficiales. 2. Los coloca en el repositorio documental del sistema. 3. El sistema procesa los documentos mediante chunking. 4. Se generan embeddings para cada fragmento. 5. Los datos son almacenados en la base vectorial FAISS.

Tabla 5 Caso de Uso UC-03: Cargar documentos normativos

3.4 DISEÑO DEL PIPELINE RAG

El pipeline RAG (Retrieval-Augmented Generation) constituye el flujo central de funcionamiento del prototipo, ya que integra dos procesos principales; recuperación semántica de información normativa y la generación de respuestas mediante un modelo de lenguaje. Su diseño busca asegurar que las respuestas del asistente se fundamenten exclusivamente en normativa institucional de la PUCE previamente cargada en la base de conocimiento, no en un conocimiento general o común como los LLMs actuales, garantizando así la coherencia y confiabilidad del sistema.

3.4.1 Pipeline de construcción de la base de conocimiento

Esta etapa se ejecuta antes de que el usuario realice consultas. Su objetivo es preparar la normativa institucional para que pueda ser recuperada eficientemente por similitud semántica.

Paso 1: Recolección y selección de normativa

Se recopilan documentos normativos institucionales de la PUCE (reglamentos, políticas, estatutos y lineamientos) en formato digital. La selección se realiza conforme al alcance del proyecto, evitando documentación no normativa.

Paso 2: Conversión y normalización de texto

Los documentos seleccionados se convierten a texto y se normalizan para reducir ruido y asegurar consistencia. Este proceso incluye la eliminación de elementos no relevantes para la consulta (por ejemplo, encabezados repetitivos, numeración de páginas o secciones redundantes).

Paso 3: Segmentación (chunking)

El texto normativo se divide en fragmentos (chunks) para facilitar su almacenamiento y recuperación. La segmentación se realiza procurando que cada fragmento represente una unidad semántica coherente (por ejemplo, un artículo, inciso o sección), de manera que el contexto recuperado sea útil para responder preguntas.

Cada fragmento se almacena junto con metadatos mínimos, tales como:

- Título del documento normativo
- Sección o artículo de referencia
- Identificador del fragmento

Paso 4: Generación de embeddings

Cada fragmento se transforma en un embedding (vector numérico) mediante un modelo de embeddings de texto. Esta representación captura el significado del contenido y permite comparar fragmentos con consultas en lenguaje natural.

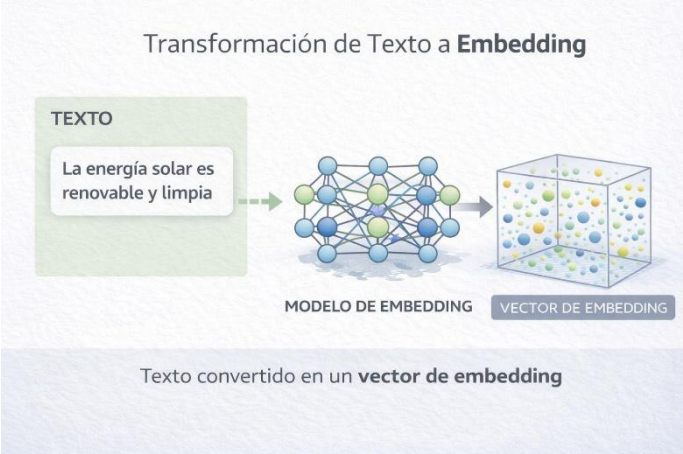

Figura 7 *Transformación a Embedding*

Fuente: Elaboración propia a través de DALL-E 3

Paso 5: Almacenamiento en base de datos vectorial

Finalmente, los embeddings y su información asociada (texto original + metadatos) se almacenan en una base de datos vectorial. Esto permite realizar búsquedas por similitud de manera eficiente durante la etapa de consulta.

Figura 8 *Almacenamiento en Base de Datos Vectorial*

Fuente: Elaboración propia a través de DALL-E 3

3.4.2 Pipeline de consulta y generación de respuestas

Esta etapa se ejecuta cuando un usuario interactúa con el asistente. Su objetivo es recuperar la normativa relevante y generar una respuesta basada únicamente en dicha información.

Paso 1: Ingreso de consulta del usuario

El usuario ingresa una pregunta o solicitud en lenguaje natural a través de la interfaz del sistema. Ejemplos:

- "¿Qué establece el reglamento sobre asistencia mínima?"
- "¿Cuáles son las condiciones para una sanción académica?"

Paso 2: Normalización de la consulta

La consulta se prepara para el proceso de búsqueda, aplicando limpieza básica (por ejemplo, eliminación de caracteres inválidos) para garantizar un procesamiento uniforme.

Paso 3: Vectorización de la consulta

La consulta se transforma en un embedding. Esto garantiza que tanto los documentos como las consultas se encuentren en el mismo espacio vectorial.

Paso 4: Recuperación semántica (retrieval)

El embedding de la consulta se compara con los embeddings almacenados en la base vectorial. El sistema recupera los k fragmentos más relevantes (top-k), según la similitud semántica.

En esta fase, el resultado es un conjunto de fragmentos normativos (con metadatos) que contienen información potencialmente útil para responder la consulta.

Paso 5: Construcción del contexto

Los fragmentos recuperados se organizan y estructuran como contexto de entrada para el modelo de lenguaje.

Paso 6: Generación de respuesta con el modelo de lenguaje

El contexto recuperado y la consulta del usuario se envían al modelo de lenguaje mediante la API. El modelo genera una respuesta natural, clara y coherente, utilizando exclusivamente el contenido proporcionado como contexto.

Paso 7: Entrega de respuesta al usuario

El sistema presenta la respuesta final en la interfaz.

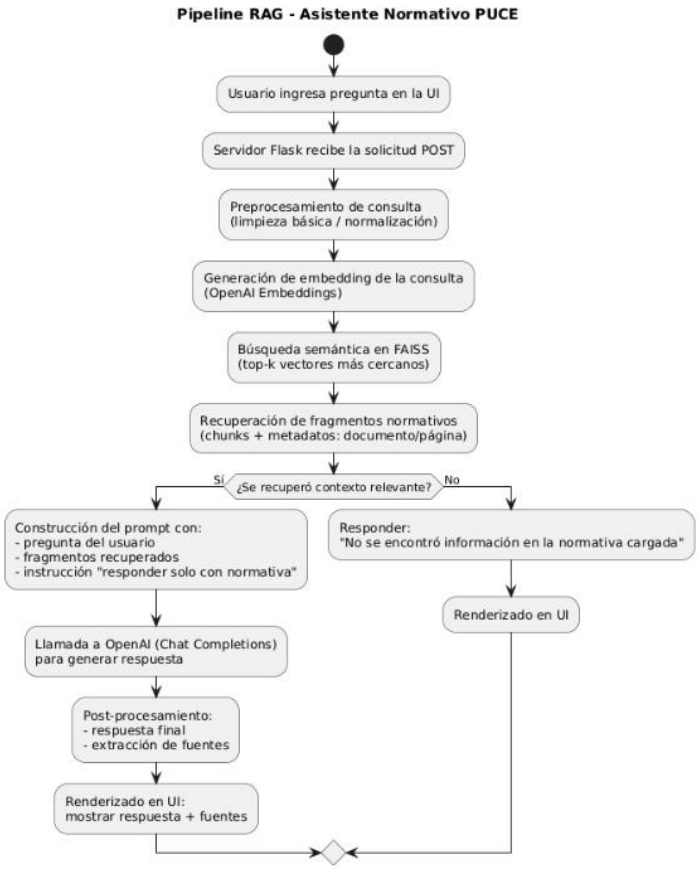

Figura 9 Diagrama Pipeline RAG

Fuente: Elaboración propia a través de PlantUML

3.5 DISEÑO DE LA BASE DE DATOS VECTORIAL

La base de datos vectorial constituye el componente central del sistema encargado de almacenar las representaciones semánticas (embeddings) generadas a partir de los fragmentos de normativa institucional. Su función principal es permitir la recuperación eficiente de información relevante mediante búsquedas por similitud, lo cual resulta en esencial la arquitectura RAG.

A diferencia de una base de datos relacional tradicional, una base vectorial no se centra únicamente en búsquedas exactas por palabras clave, sino en la comparación matemática de vectores de alta dimensión, lo cual posibilita recuperar fragmentos normativos relacionados conceptualmente con la consulta del usuario.

3.5.1 Propósito y alcance de la base vectorial

La base de datos vectorial del prototipo fue diseñada para:

- Almacenar embeddings asociados a fragmentos normativos de la PUCE.
- Mantener el texto original de cada fragmento y sus metadatos.
- Permitir búsquedas semánticas (top-k) a partir de una consulta del usuario.
- Servir como fuente controlada de contexto para la generación de respuestas mediante el enfoque RAG.

En concordancia con el alcance del proyecto, la base vectorial se alimenta exclusivamente con documentos normativos institucionales públicos.

3.5.2 Unidad de almacenamiento (documento y fragmento)

Para optimizar la recuperación semántica, la normativa institucional se almacena en unidades denominadas fragmentos normativos. Cada fragmento corresponde a una porción de texto coherente (por ejemplo, un artículo, inciso o sección), con el objetivo de:

- Reducir ruido en la recuperación de información.
- Facilitar la trazabilidad hacia la fuente normativa.
- Proporcionar contexto específico al modelo de lenguaje.

Por lo tanto, el elemento principal almacenado en la base vectorial es:

- Embedding del fragmento (vector numérico)
- Texto del fragmento (contenido original)
- Metadatos del fragmento (para referencia y trazabilidad)

3.5.3 Proceso de indexación vectorial

El proceso de indexación inicia una vez que se generan embeddings, los pasos a seguir son:

1. Se genera el embedding del fragmento normativo.
2. Se asigna un identificador único.
3. Se almacenan embedding, texto y metadatos en la base vectorial.
4. Se habilita la búsqueda por similitud mediante un índice vectorial.

Este índice es la estructura que permite recuperar eficientemente los fragmentos más cercanos a una consulta, sin necesidad de recorrer todos los registros de forma manual.

3.5.4 Consulta y recuperación (búsqueda top-k)

Cuando el usuario realiza una consulta:

1. La consulta se transforma en embedding.
2. La base vectorial calcula la similitud entre el embedding de consulta y los embeddings almacenados.
3. Se recuperan los k fragmentos más similares (top-k).
4. Los fragmentos recuperados se envían al módulo generador como contexto.

La selección de top-k permite equilibrar dos factores:

- **Precisión:** menos fragmentos pueden ser más relevantes.
- **Cobertura:** más fragmentos pueden cubrir distintas partes de la normativa asociada.

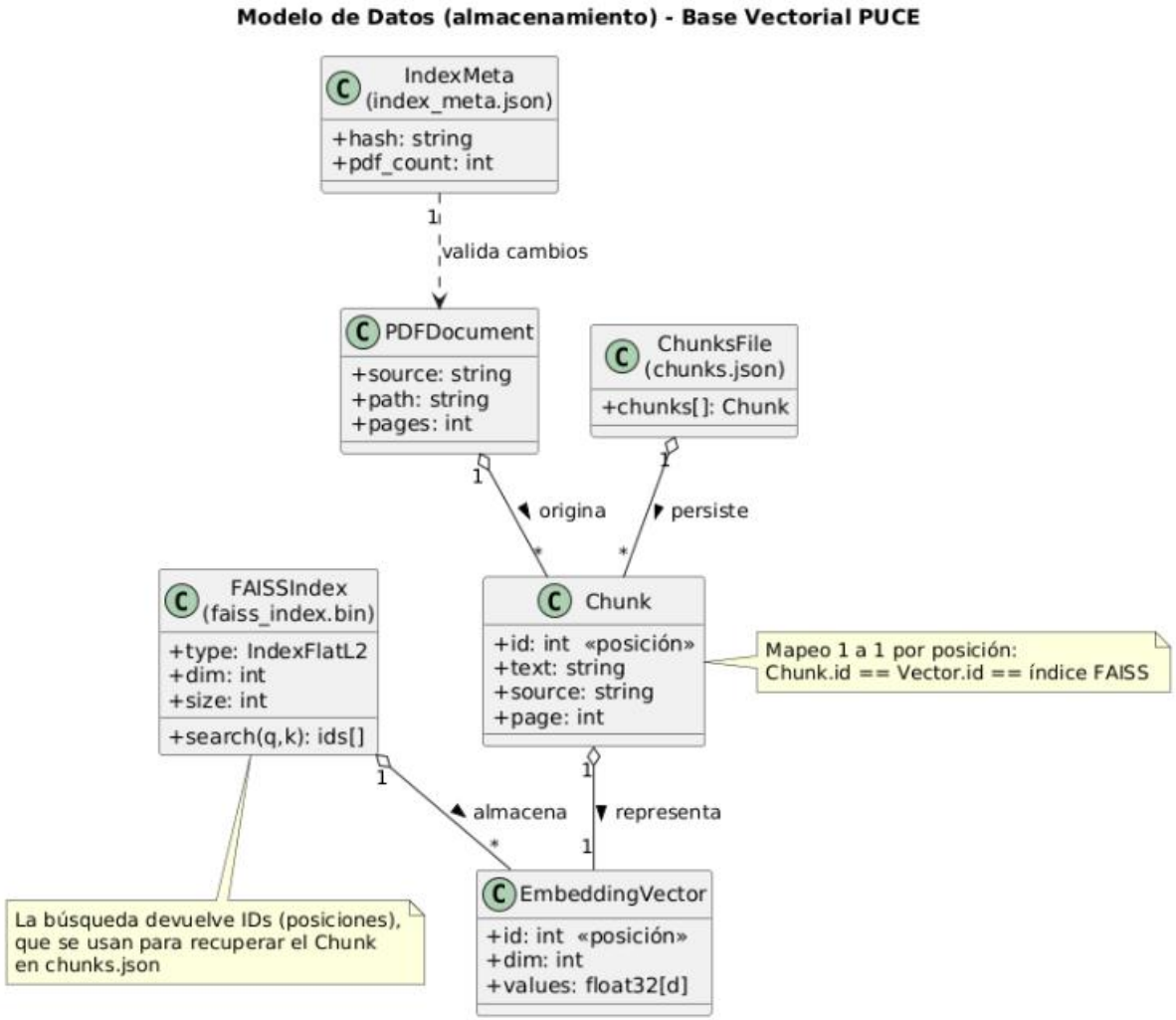

Figura 10 *Diseño Base Vectorial*

Fuente: Elaboración propia a través de PlantUML

3.6 DISEÑO DE LA INTERFAZ DEL USUARIO (UI)

El diseño de la interfaz del usuario (UI) tiene como objetivo facilitar la interacción entre el usuario y el prototipo de asistente de inteligencia artificial, permitiendo la consulta eficiente de normativa institucional de la Pontificia Universidad Católica del Ecuador (PUCE).

La UI actúa como el punto de entrada y salida del sistema, permitiendo al usuario formular consultas en lenguaje natural y visualizar las respuestas generadas por el asistente de manera comprensible y estructurada.

3.6.1 Principios de diseño de la interfaz

El diseño de la interfaz se fundamenta en los siguientes principios:

- **Simplicidad:** minimizar elementos visuales innecesarios para centrar la atención en la consulta y la respuesta.
- **Claridad:** presentar la información de forma ordenada y legible.
- **Accesibilidad:** permitir el uso del sistema desde dispositivos comunes sin requerir configuraciones especiales.
- **Consistencia:** mantener una estructura uniforme en todas las interacciones.
- **Orientación al usuario:** priorizar la experiencia del estudiante en la consulta normativa.

Figura 11 *Ejemplo de UI*

Fuente: Pantalla Consulta ChatGPT.com

3.6.2 Componentes principales de la interfaz

La interfaz del prototipo se estructura en los siguientes componentes:

a) Área de ingreso de consulta

Espacio destinado para que el usuario ingrese su pregunta en lenguaje natural.

a. Ejemplo de consulta:
"¿Qué establece el reglamento sobre asistencia mínima?"

b) Botón de envío

Elemento que permite enviar la consulta al sistema para su procesamiento. Al activarse, se inicia el flujo de recuperación semántica y generación de respuesta.

c) Área de visualización de la respuesta

Sección donde se presenta la respuesta generada por el asistente. La respuesta se muestra en lenguaje claro y estructurado.

3.6.3 Flujo de interacción del usuario

El flujo de interacción con la interfaz se desarrolla de la siguiente manera:

1. El usuario accede a la interfaz del asistente.
2. Ingresa una consulta en lenguaje natural.
3. Envía la consulta mediante el botón correspondiente.
4. El sistema procesa la consulta a través del pipeline RAG.
5. La respuesta generada se presenta en el área de resultados.
6. El usuario puede realizar nuevas consultas de manera consecutiva.

3.6.4 Alcance del diseño de la interfaz

El diseño de la interfaz se limita a las funcionalidades necesarias para la validación del prototipo, sin incluir características avanzadas como autenticación de usuarios, personalización de perfiles o historial de consultas. Estas funcionalidades se consideran fuera del alcance del proyecto y se proponen como posibles mejoras futuras.

CAPITULO IV: DESARROLLO E IMPLEMENTACIÓN DEL PROTOTIPO

Este capítulo describe el proceso de desarrollo e implementación del prototipo de asistente de inteligencia artificial propuesto en la investigación. Se detallan las tecnologías utilizadas, el entorno de desarrollo, la organización del trabajo por iteraciones y la implementación de los principales módulos del sistema, evidenciando la construcción progresiva del prototipo y su funcionamiento conforme a la arquitectura definida en el capítulo anterior.

4.1 Tecnologías y herramientas utilizadas

Para el desarrollo e implementación del prototipo de asistente de inteligencia artificial se seleccionaron tecnologías y herramientas ampliamente utilizadas en proyectos de inteligencia artificial y desarrollo de software, tomando en cuenta su estabilidad, disponibilidad en entornos académicos y compatibilidad con el enfoque de Recuperación Aumentada por Generación (RAG).

4.1.1 Lenguaje de programación

- **Python**
  Se utilizó Python como lenguaje principal de desarrollo debido a su amplia adopción en proyectos de inteligencia artificial, su extenso ecosistema de librerías para procesamiento de lenguaje natural y su facilidad de integración con servicios externos y bases de datos vectoriales.
    - **Flask**
      La interfaz fue desarrollada utilizando plantillas HTML renderizadas por Flask, con apoyo del motor de plantillas Jinja2

4.1.2 Editor de texto / Entorno de desarrollo integrado

- **Visual Studio Code (VS Code)**
  Se empleó Visual Studio Code como editor de texto principal, debido a su soporte para Python, gestión de entornos virtuales, integración con control de versiones y extensiones especializadas para desarrollo de aplicaciones de inteligencia artificial.

4.1.3 Modelo de lenguaje

- **API de ChatGPT (OpenAI)**

Se utilizó la API de ChatGPT como modelo de lenguaje de gran escala para la generación de respuestas en lenguaje natural, integrándola mediante el enfoque RAG para asegurar que las respuestas estén basadas exclusivamente en normativa institucional.

4.1.4 Modelo de embeddings

- **Modelo de embeddings de texto de OpenAI**
  Se empleó un modelo de embeddings de texto de la API de OpenAI para transformar fragmentos normativos y consultas del usuario en representaciones vectoriales, posibilitando la búsqueda semántica eficiente.

4.1.5 Base de datos vectorial

- **FAISS (Facebook AI Similarity Search)**,
  Para el almacenamiento y recuperación semántica de la normativa institucional se utilizó una base de datos vectorial implementada mediante la librería FAISS (Facebook AI Similarity Search), la cual permite indexar embeddings de texto y realizar búsquedas por similitud semántica sin necesidad de un servidor externo.

4.1.6 Procesamiento de documentos

- **Librerías de extracción y procesamiento de texto**
  Se utilizaron herramientas de procesamiento de documentos en Python para convertir archivos normativos a texto plano, facilitar su limpieza, segmentación y posterior análisis semántico;

    - **pypdf:** Extrae texto directamente desde PDFs normativos
    - **pdfplumber:** Extrae texto directamente desde PDFs normativos, pero funciona mejor cuando hay tablas o formatos complejos
    - **re:** Para limpiar encabezados, espacios, numeración repetida.

- **langchain:** Ideal para dividir documentos en fragmentos normativos correctamente.
- **openai:** Librería propia de OpenAI

4.1.7 Control de versiones

- **Git + GitHub**

  Para el control de versiones del prototipo se utilizó Git como sistema de gestión de cambios, alojando el proyecto en un repositorio de GitHub, lo cual permitió mantener trazabilidad del desarrollo, organización del código fuente y respaldo de iteraciones realizadas durante el proceso de implementación.

4.2 ORGANIZACIÓN DEL DESARROLLO POR ITERACIONES

El desarrollo del prototipo se realizó siguiendo una metodología iterativa–incremental, permitiendo construir el sistema de manera progresiva y realizar ajustes continuos en función de los resultados obtenidos.

El trabajo se organizó en las siguientes iteraciones:

4.2.1 Iteración 1: Preparación del entorno y base de conocimiento

- Configuración del entorno de desarrollo y editor de texto.
- Organización de la estructura del proyecto.
- Recolección y procesamiento inicial de normativa institucional.
- Generación y almacenamiento de embeddings en la base de datos vectorial.

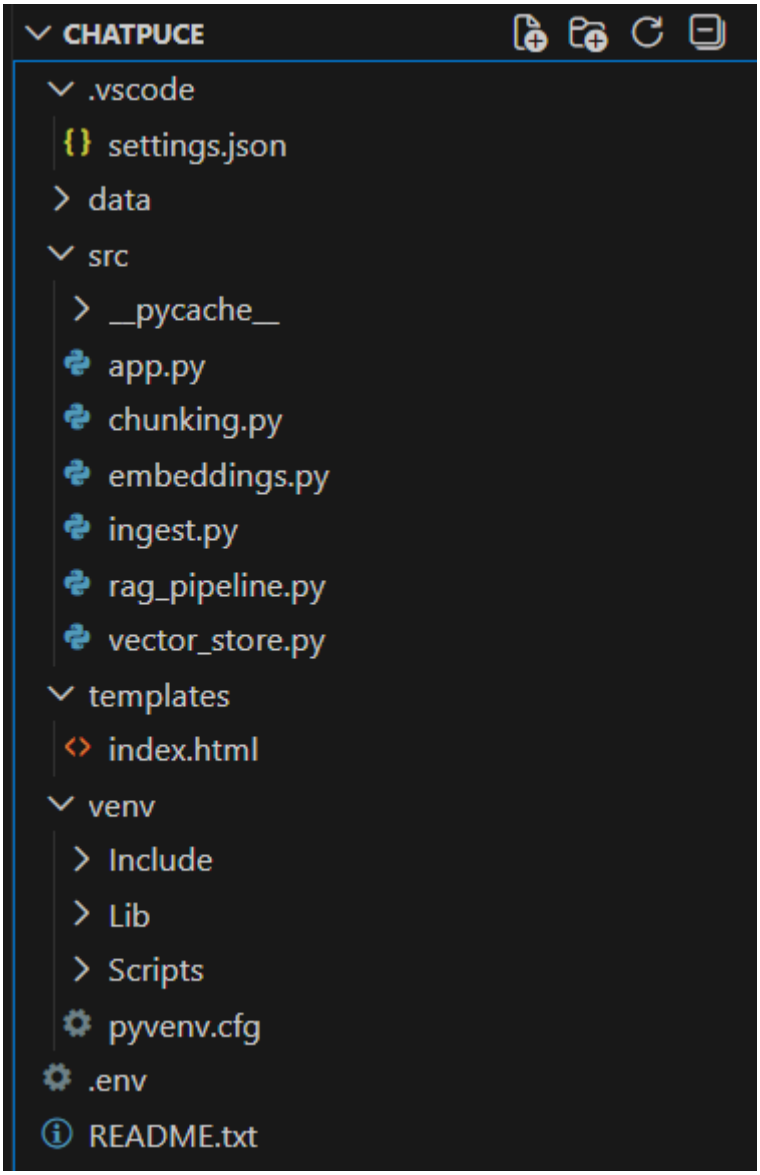

Figura 12 *Preparación de Entorno*

Fuente: Elaboración propia

4.2.2 Iteración 2: Implementación del pipeline RAG

- Implementación del módulo de recuperación semántica.
- Integración del modelo de lenguaje mediante la API correspondiente.
- Generación de respuestas basadas en normativa institucional recuperada.

```python
rag_pipeline.py > responder_pregunta
from embeddings import generar_embedding
from openai import OpenAI

client = OpenAI()

def responder_pregunta(pregunta, vectorstore):
    # 1. embedding de consulta
    emb_query = generar_embedding(pregunta)

    # 2. retrieval top-k
    fragmentos = vectorstore.search(emb_query, k=3)

    # 3. construir contexto
    contexto = "\n\n".join(fragmentos)

    # 4. generar respuesta final
    prompt = f"""
    Responde únicamente con base en normativa institucional PUCE:

    Normativa:
    {contexto}

    Pregunta:
    {pregunta}

    Respuesta:
    """

    response = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[{"role": "user", "content": prompt}]
    )

    return response.choices[0].message.content
```

Figura 13 *Fragmento de Código - Implementación del pipeline RAG*

Fuente: Elaboración propia

4.2.3 Iteración 3: Integración, interfaz y validación

- Desarrollo de la interfaz de usuario.
- Integración de todos los módulos del sistema.
- Ejecución de pruebas funcionales.
- Ajustes finales y documentación del prototipo.

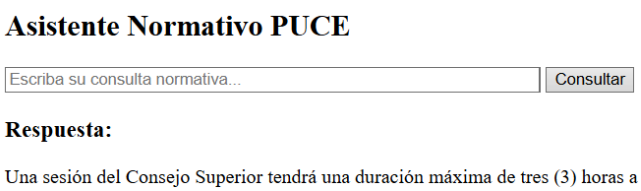

Figura 14 *Primer Prototipo y UI para pruebas*

Fuente: Elaboración propia

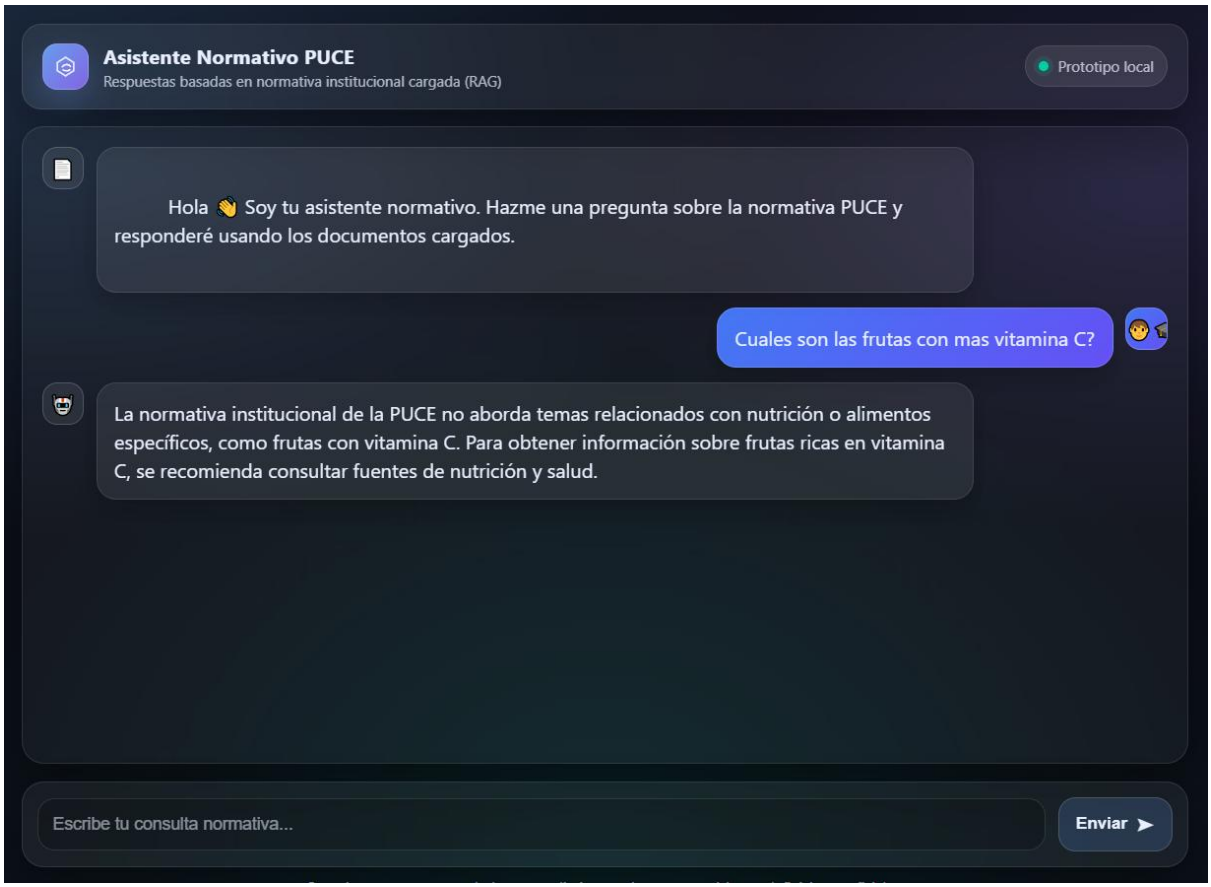

Figura 15 *Segundo Prototipo y UI para pruebas*

Fuente: Elaboración propia

Figura 16 *Tercer Prototipo y UI para pruebas*

Fuente: Elaboración propia

4.3 IMPLEMENTACIÓN DEL MÓDULO DE INGESTA DOCUMENTAL

El módulo de ingesta documental constituye la primera etapa del prototipo ya que permite incorporar la normativa institucional de la PUCE dentro de la base de conocimiento del asistente. Su función principal es transformar documentos normativos en texto procesable, segmentarlos en fragmentos semánticamente coherentes y almacenarlos en una base de datos vectorial para su posterior recuperación mediante el enfoque RAG.

4.3.1 Extracción de texto desde documentos PDF

A continuación, se muestra un ejemplo del proceso de lectura y extracción:

```python
def extraer_texto_pdf(path):
    reader = PdfReader(path)
    texto = ""

    for page in reader.pages:
        texto += page.extract_text()

    return texto
```

Figura 17 *Fragmento de Código - Extracción de texto desde documentos PDF*

Fuente: Elaboración propia

Este procedimiento permite convertir reglamentos y documentos institucionales en texto plano para su posterior procesamiento.

4.3.2 Normalización y limpieza del texto normativo

Una vez extraído el contenido, se aplicó un proceso de limpieza con el fin de reducir ruido textual, eliminando elementos repetitivos como encabezados, saltos de línea excesivos o caracteres innecesarios.

```python
4   def limpiar_texto(texto: str) -> str:
5       texto = re.sub(r"\s+", " ", texto)
6       return texto.strip()
```

Figura 18 *Fragmento de Código - Normalización y limpieza del texto normativo*

Fuente: Elaboración propia

Este paso es relevante para garantizar que la segmentación posterior sea más precisa.

4.3.3 Segmentación normativa (chunking)

Los documentos normativos son extensos y contienen múltiples artículos y disposiciones. Por ello, se implementó un mecanismo de segmentación que divide el texto en fragmentos manejables, preservando coherencia semántica.

Se utilizó el componente RecursiveCharacterTextSplitter de LangChain:

```python
def chunk_normativa(texto: str):
    splitter = RecursiveCharacterTextSplitter(
        chunk_size=500,
        chunk_overlap=50
    )
    return splitter.split_text(texto)
```

Figura 19 *Fragmento de Código - Segmentación normativa (chunking)*

Fuente: Elaboración propia

Con esta estrategia se obtienen fragmentos adecuados para ser convertidos en embeddings, manteniendo suficiente contexto normativo para responder consultas.

4.3.4 Generación de embeddings de fragmentos normativos

Posteriormente, cada fragmento segmentado se transforma en un embedding vectorial utilizando el modelo de OpenAI, permitiendo representar el significado del texto en forma numérica.

```python
client = OpenAI()

def generar_embedding(texto):
    response = client.embeddings.create(
        model="text-embedding-3-large",
        input=texto
    )
    return response.data[0].embedding
```

Figura 20 *Fragmento de Código - Generación de embeddings de fragmentos normativos*

Fuente: Elaboración propia

Estos embeddings son el insumo principal para la construcción de la base vectorial.

4.3.5 Almacenamiento e indexación en FAISS

Finalmente, los embeddings generados se almacenan en una base de datos vectorial implementada con FAISS, permitiendo realizar búsquedas semánticas eficientes durante la fase de consulta.

```python
class VectorStoreFAISS:
    def __init__(self, dim):
        self.index = faiss.IndexFlatL2(dim)
        self.textos = []

    def add(self, embedding, fragmento):
        self.index.add(np.array([embedding]).astype("float32"))
        self.textos.append(fragmento)

    def search(self, embedding_query, k=3):
        D, I = self.index.search(
            np.array([embedding_query]).astype("float32"),
            k
        )
        return [self.textos[i] for i in I[0]]
```

Figura 21 *Fragmento de Código - Almacenamiento e indexación en FAISS*

Fuente: Elaboración propia

De esta forma, la normativa institucional queda indexada para ser recuperada por similitud semántica frente a consultas del usuario.

4.4 IMPLEMENTACIÓN DE LA RECUPERACIÓN SEMÁNTICA Y GENERACIÓN DE RESPUESTAS

Una vez construida la base de conocimiento normativa mediante el módulo de ingesta documental, el siguiente paso en el desarrollo del prototipo consiste en implementar el proceso de consulta en tiempo real, basado en la arquitectura Retrieval-Augmented Generation (RAG).

Este módulo permite que el asistente responda preguntas del usuario utilizando únicamente fragmentos normativos recuperados desde la base vectorial.

La implementación de este componente se desarrolla en cuatro etapas principales:

1. Vectorización de la consulta del usuario
2. Recuperación semántica de fragmentos relevantes
3. Construcción del contexto normativo
4. Generación de respuesta mediante el modelo de lenguaje

4.4.1 Vectorización de la consulta del usuario

Cuando el usuario realiza una consulta en lenguaje natural, dicha consulta debe transformarse en un embedding para poder compararse con los embeddings almacenados en la base vectorial.

```python
def responder_pregunta(pregunta, vectorstore):
    # 1. embedding de consulta
    emb_query = generar_embedding(pregunta)
```

```python
def generar_embedding(texto):
    response = client.embeddings.create(
        model="text-embedding-3-large",
        input=texto
    )
    return response.data[0].embedding
```

Figura 22 *Fragmento de Código - Vectorización de la consulta del usuario*

Fuente: Elaboración propia

De esta manera, la consulta y los fragmentos normativos se encuentran en el mismo espacio vectorial, habilitando la recuperación semántica.

4.4.2 Recuperación de fragmentos normativos mediante FAISS

Una vez obtenido el embedding de la consulta, se realiza una búsqueda top-k en el índice FAISS para recuperar los fragmentos más relevantes según similitud semántica.

```python
class VectorStoreFAISS:
    def __init__(self, dim):
        self.index = faiss.IndexFlatL2(dim)
        self.textos = []

    def add(self, embedding, fragmento):
        self.index.add(np.array([embedding]).astype("float32"))
        self.textos.append(fragmento)

    def search(self, embedding_query, k=3):
        D, I = self.index.search(
            np.array([embedding_query]).astype("float32"),
            k
        )
        return [self.textos[i] for i in I[0]]
```

Figura 23 *Fragmento de Código - Recuperación de fragmentos normativos mediante FAISS*

Fuente: Elaboración propia

Este proceso devuelve los fragmentos normativos más cercanos conceptualmente a la consulta realizada.

4.4.3 Construcción del contexto normativo

Los fragmentos recuperados deben organizarse en un contexto estructurado que será enviado al modelo de lenguaje como base documental.

```
# 3. construir contexto
contexto = "\n\n".join(fragmentos)
```

```
  def search(self, embedding_query, k=3):
      D, I = self.index.search(
          np.array([embedding_query]).astype("float32"),
          k
      )
      return [self.textos[i] for i in I[0]]
```

Figura 24 *Fragmento de Código - Construcción del contexto normativo*

Fuente: Elaboración propia

El contexto resultante contiene únicamente normativa institucional de la PUCE, asegurando que el modelo no genere respuestas basadas en información externa o no verificada.

4.4.4 Generación de respuesta con ChatGPT mediante API

Finalmente, la consulta del usuario junto con el contexto normativo se envía al modelo de lenguaje mediante la API de OpenAI, solicitando que la respuesta se base exclusivamente en el contenido proporcionado.

```python
def responder_pregunta(pregunta, vectorstore):
    # 1. embedding de consulta
    emb_query = generar_embedding(pregunta)

    # 2. retrieval top-k
    fragmentos = vectorstore.search(emb_query, k=3)

    # 3. construir contexto
    contexto = "\n\n".join(fragmentos)

    # 4. generar respuesta final
    prompt = f"""
    Responde únicamente con base en normativa institucional PUCE:

    Normativa:
    {contexto}

    Pregunta:
    {pregunta}

    Respuesta:
    """

    response = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[{"role": "user", "content": prompt}]
    )

    return response.choices[0].message.content
```

Figura 25 *Fragmento de Código - Generación de respuesta con ChatGPT mediante API*

Fuente: Captura de Pantalla - Elaboración propia

Este enfoque garantiza que el asistente actúe como un sistema experto normativo, evitando respuestas inventadas o no alineadas con la base documental.

4.5 Implementación de la interfaz del usuario e integración final

Una vez implementados los módulos principales del sistema se desarrolló una interfaz de usuario que permite la interacción directa con el asistente normativo. La interfaz constituye el componente final del prototipo, ya que actúa como medio de comunicación entre el usuario y

el sistema, facilitando el ingreso de consultas en lenguaje natural y la visualización de las respuestas generadas a partir de normativa institucional de la PUCE.

4.5.1 Interfaz del usuario

La interfaz del usuario fue implementada con una estructura sencilla, como se muestra a continuación:

```
    {% if pregunta %}
    <div class="msg user">
      <div class="bubble">{{ pregunta }}</div>
      <div class="avatar user" aria-hidden="true">🧑‍🎓</div>
    </div>
    {% endif %}

    {% if respuesta %}
    <div class="msg bot">
      <div class="avatar" aria-hidden="true">🤖</div>
      <div class="bubble">{{ respuesta }}</div>
    </div>

    {% if fuentes and fuentes|length > 0 %}
    <div class="meta">
      <h3>Fuentes utilizadas</h3>
      <div class="chips">
        {% for f in fuentes %}
          <span class="chip">{{ f }}</span>
        {% endfor %}
      </div>
    </div>
    {% endif %}
    {% endif %}

    <div class="typing" id="typing">
      <span>Generando respuesta</span>
      <span class="b"></span><span class="b"></span><span class="b"></span>
    </div>
  </main>
```

Figura 26 *Fragmento de código UI*

Fuente: Captura de Pantalla - Elaboración propia

Esta interfaz permite realizar consultas consecutivas y visualizar respuestas generadas por el modelo.

CAPÍTULO V: PRUEBAS Y RESULTADOS

El presente capítulo describe el proceso de pruebas y validación del prototipo desarrollado, así como los resultados obtenidos durante su funcionamiento. La finalidad de esta fase es verificar que el asistente de inteligencia artificial cumple con los requerimientos establecidos, responde adecuadamente a consultas normativas y demuestra la viabilidad técnica del enfoque basado en Recuperación Aumentada por Generación (RAG).

Para ello, se aplicaron pruebas funcionales orientadas a evaluar el comportamiento del sistema en escenarios reales de consulta, considerando la normativa institucional de la Pontificia Universidad Católica del Ecuador (PUCE) como única fuente documental integrada en la base de conocimiento.

5.1 Plan de pruebas

El plan de pruebas constituye una estrategia sistemática para verificar el correcto funcionamiento del prototipo. Este plan se diseñó en función de los requerimientos funcionales y no funcionales definidos en el Capítulo IV.

Las pruebas realizadas se enfocaron en los siguientes aspectos:

- Correcta recuperación semántica de fragmentos normativos.
- Generación de respuestas basadas únicamente en normativa PUCE.
- Funcionamiento general de la interfaz de consulta.
- Tiempo de respuesta aceptable en un entorno académico local.

5.2 Pruebas funcionales del sistema

Las pruebas funcionales consistieron en ejecutar consultas representativas sobre normativa institucional, evaluando que el sistema:

- Interprete correctamente la pregunta del usuario.
- Recupere fragmentos normativos relevantes mediante FAISS.
- Genere una respuesta coherente basada en el contexto recuperado.
- Mantenga consistencia con el alcance definido (solo normativa).

Las pruebas fueron realizadas en un entorno local utilizando documentos normativos previamente cargados en la base vectorial.

Figura 27 *Prueba 1 – Funcionamiento del Sistema*

Fuente: Elaboración propia

Pregunta	Respuesta
¿Como se maneja la asistencia?	La asistencia a clases y actividades académicas en la PUCE es un acto autónomo y responsable de los estudiantes. Los profesores llevarán un registro de asistencia, el cual es importante para el seguimiento académico del estudiante y para el proceso de evaluación permanente. La asistencia a actividades prácticas estará regulada por normas específicas. En caso de ausencias, estas podrán ser recuperadas según lo establecido en los artículos 43 y 44 del Reglamento General de Estudiantes. Es importante notar que la inasistencia no será causal de pérdida de las asignaturas. Sin embargo, cuando se realicen actividades prácticas o se apliquen metodologías activas, la asistencia puede ser considerada como parte de un criterio de evaluación, que debe estar claramente indicado en el sílabo, siguiendo los lineamientos del Vicerrectorado de Docencia y Estudiantes. Para aprobar una asignatura, el estudiante debe obtener al menos el 60% de la nota en todos los resultados de aprendizaje.

Tabla 6 *Prueba 1 – Resultado*

Figura 28 *Prueba 2 – Funcionamiento del Sistema*

Fuente: Elaboración propia

Pregunta	Respuesta
¿Cuáles son las frutas con más vitamina C?	Lo siento, pero no puedo responder a esa pregunta ya que no se relaciona con la normativa institucional de la PUCE. Si tienes preguntas sobre los requisitos para títulos de cuarto nivel o sobre las becas, estaré encantado de ayudarte

Tabla 7 *Prueba 2 – Resultado*

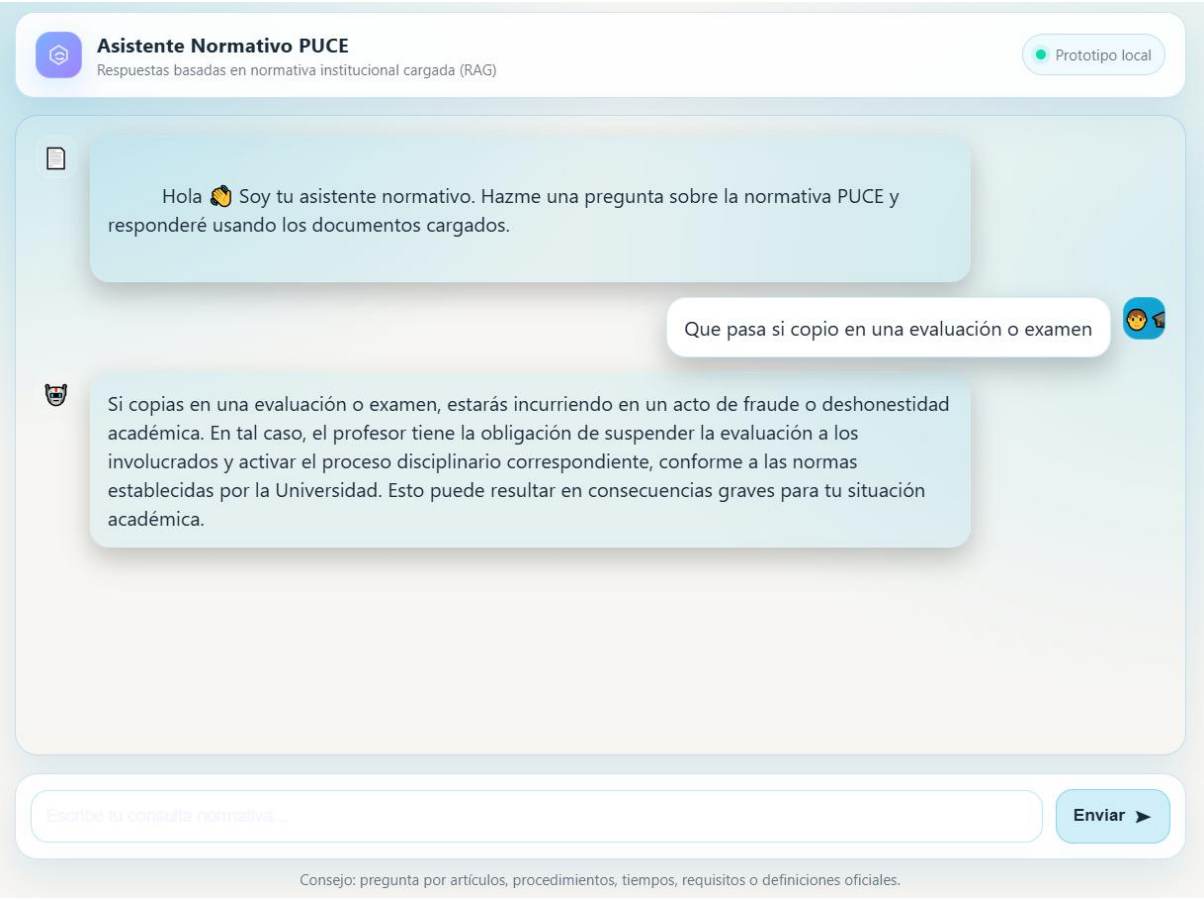

Figura 29 *Prueba 3 – Funcionamiento del Sistema*

Fuente: Elaboración propia

Pregunta	Respuesta
¿Qué pasa si copio en una evaluación o examen?	Lo siento, pero no puedo responder a esa pregunta ya que no se relaciona con la normativa institucional de la PUCE. Si tienes preguntas sobre los requisitos para títulos de cuarto nivel o sobre las becas, estaré encantado de ayudarte

Tabla 8 *Prueba 3 – Resultado*

5.3 Casos de prueba aplicados

A continuación, se presentan algunos casos de prueba ejecutados durante la validación del prototipo:

Caso	Consulta realizada	Resultado esperado	Resultado obtenido
CP-01	¿Como se maneja la asistencia?	La asistencia a clases y actividades académicas en la PUCE es un acto autónomo y responsable de los estudiantes. Los profesores llevarán un registro de asistencia, el cual es importante para el seguimiento académico del estudiante y para el proceso de evaluación permanente. La asistencia a actividades prácticas estará regulada por normas específicas. En caso de ausencias, estas podrán ser recuperadas según lo	La asistencia a clases y actividades académicas en la PUCE es un acto autónomo y responsable de los estudiantes. Los profesores llevarán un registro de asistencia, el cual es importante para el seguimiento académico del estudiante y para el proceso de evaluación permanente. La asistencia a actividades prácticas estará regulada por normas específicas. En caso de ausencias, estas podrán ser recuperadas según lo

		establecido en los artículos 43 y 44 del Reglamento General de Estudiantes. Es importante notar que la inasistencia no será causal de pérdida de las asignaturas. Sin embargo, cuando se realicen actividades prácticas o se apliquen metodologías activas, la asistencia puede ser considerada como parte de un criterio de evaluación, que debe estar claramente indicado en el sílabo, siguiendo los lineamientos del Vicerrectorado de Docencia y Estudiantes. Para aprobar una asignatura, el estudiante debe obtener al menos el 60% de la nota en todos los resultados de aprendizaje.	establecido en los artículos 43 y 44 del Reglamento General de Estudiantes. Es importante notar que la inasistencia no será causal de pérdida de las asignaturas. Sin embargo, cuando se realicen actividades prácticas o se apliquen metodologías activas, la asistencia puede ser considerada como parte de un criterio de evaluación, que debe estar claramente indicado en el sílabo, siguiendo los lineamientos del Vicerrectorado de Docencia y Estudiantes. Para aprobar una asignatura, el estudiante debe obtener al menos el 60% de la nota en todos los resultados de aprendizaje.
CP-02	¿Cuáles son las frutas con más vitamina C?	Lo siento, pero no puedo responder a esa pregunta ya que no se relaciona con la normativa institucional de la PUCE. Si tienes preguntas sobre los requisitos para títulos de cuarto nivel o sobre las becas, estaré encantado de ayudarte	Solo poseo información relacionada con la normativa institucional de la PUCE. No tengo información relacionada con frutas o vitamina C.
CP-03	¿Qué pasa si copio en una evaluación o examen?	Si copias en una evaluación o examen, estarás incurriendo en un acto de fraude o deshonestidad académica. En tal caso, el profesor tiene la obligación de suspender la evaluación a los	En caso de fraude o deshonestidad académica detectada in fraganti el profesor deberá suspender la evaluación a los involucrados y activará el proceso disciplinario

	involucrados y activar el proceso disciplinario correspondiente, conforme a las normas establecidas por la Universidad. Esto puede resultar en consecuencias graves para tu situación académica.	correspondiente conforme a las normas de la Universidad. Artículo reformado por resolución CS-15-5 adoptada por el Consejo Superior de la PUCE el 23 de julio de 2025.

Tabla 9 *Comparativa Resultado Esperado – Resultado Obtenido*

Estos casos evidencian que el prototipo responde de forma apropiada ante consultas normativas comunes dentro de la comunidad universitaria.

5.4 Resultados obtenidos

Los resultados generales del prototipo evidencian que el sistema logra:

- Recuperar información relevante desde documentos normativos institucionales.
- Generar respuestas comprensibles para estudiantes sin conocimientos técnicos.
- Reducir el riesgo de respuestas incorrectas mediante el uso del enfoque RAG.
- Operar adecuadamente como prototipo académico en entorno local.

Asimismo, se observó que la calidad de las respuestas depende directamente de:

- La correcta segmentación normativa.
- La actualización de los documentos cargados.
- La cantidad de fragmentos recuperados (top-k).

5.5 Validación del cumplimiento de requerimientos

Con el fin de verificar el cumplimiento de los requerimientos establecidos, se evaluaron los principales requerimientos funcionales mediante pruebas directas:

Código	Requerimiento	Validación
RF-05	Consulta en lenguaje natural	Cumplido mediante interfaz funcional
RF-06	Recuperación semántica	Cumplido mediante FAISS top-k
RF-07	Respuesta basada en normativa	Cumplido mediante contexto RAG
RF-08	Respuesta clara para estudiantes	Cumplido en casos evaluados
RNF-02	Protección de información sensible	Cumplido, solo normativa pública

Tabla 10 *Validación cumplimiento de requerimientos*

De esta manera, se confirma que el prototipo cumple los requerimientos esenciales definidos en el diseño del sistema.

5.6 Discusión y limitaciones de la validación

Si bien los resultados demuestran la viabilidad del prototipo, es importante considerar que:

- La validación se realizó con un conjunto limitado de normativas seleccionadas.
- El sistema no incorpora actualización automática de documentos.
- La precisión de respuesta depende del contenido disponible en la base vectorial.

Estas limitaciones se encuentran alineadas con el alcance académico del proyecto y pueden considerarse como oportunidades de mejora futura.

CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

El presente capítulo expone las conclusiones obtenidas a partir del desarrollo e implementación del prototipo de asistente de inteligencia artificial orientado a la consulta de normativa institucional de la Pontificia Universidad Católica del Ecuador (PUCE).

6.1 Conclusiones

A partir del desarrollo del presente proyecto de titulación, se concluye que el uso de técnicas modernas de inteligencia artificial aplicadas a la gestión documental representa una alternativa viable y pertinente para facilitar el acceso a normativa institucional dentro de entornos universitarios.

En primer lugar, se logró desarrollar un prototipo funcional de asistente de inteligencia artificial basado en el enfoque de Recuperación Aumentada por Generación (RAG), el cual permite responder consultas en lenguaje natural utilizando exclusivamente documentos normativos institucionales previamente incorporados en una base de conocimiento.

Asimismo, se evidenció que la utilización de embeddings semánticos y bases vectoriales permite superar las limitaciones de los sistemas tradicionales de búsqueda por palabras clave, logrando recuperar fragmentos relevantes de normativa incluso cuando la consulta del usuario no coincide de manera literal con el texto original.

Por otra parte, las pruebas funcionales aplicadas demostraron que el prototipo opera correctamente en un entorno académico local, recuperando información normativa de manera coherente y generando respuestas claras y comprensibles para usuarios sin conocimientos técnicos avanzados.

Finalmente, se concluye que un asistente normativo basado en inteligencia artificial puede fortalecer los procesos de gestión documental institucional, promoviendo un acceso más rápido, organizado y contextualizado a la normativa oficial universitaria.

6.2 Recomendaciones

En función de los resultados obtenidos, se proponen las siguientes recomendaciones:

- Se recomienda ampliar progresivamente la base documental normativa incorporada en el sistema, integrando nuevas versiones y documentos oficiales para mejorar la cobertura de consultas.
- Se recomienda incorporar referencias explícitas en las respuestas generadas, tales como artículos o secciones normativas específicas, fortaleciendo la trazabilidad y confiabilidad del sistema.
- Se plantea como mejora futura el desarrollo de una interfaz más robusta e institucional, con funcionalidades adicionales como historial de consultas, categorización temática y búsqueda avanzada.
- Se recomienda realizar pruebas con una muestra más amplia de estudiantes y personal administrativo para evaluar el impacto del asistente en un entorno universitario real.

REFERENCIAS

What are Vector Embeddings. (s. f.). Pinecone. https://www.pinecone.io/learn/vector-embeddings

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., … & Rocktäschel, T. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Proceedings of NeurIPS 2020.

Myers, J. (2024, 23 septiembre). OpenAI Text Embedding Models: A Beginner's Guide. The New Stack. https://thenewstack.io/beginners-guide-to-openai-text-embedding-models/?utm_source=chatgpt.com

Merritt, R. (2025, 9 octubre). What Is Retrieval-Augmented Generation aka RAG | NVIDIA Blogs. NVIDIA Blog. https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/?utm_source=chatgpt.com

EvoAcademy. (2024, 18 octubre). *¿Qué es RAG? - Clase con código y ejemplo* [Vídeo]. YouTube. https://www.youtube.com/watch?v=uAsd9pOIcLg

NetMentor. (2025, 28 octubre). *Guía Completa: Cómo Crear un ChatGPT privado con tu Propia Información usando RAG y C#* [Vídeo]. YouTube. https://www.youtube.com/watch?v=ZpHCDjDVPpg

PlantUML. (s. f.). *herramienta de código abierto que utiliza descripciones textuales simples para dibujar hermosos diagramas UML.* PlantUML.com. https://plantuml.com/es/

Lucidchart | Diagramas creados con inteligencia. (s. f.). Lucidchart. https://www.lucidchart.com/pages/es/landing?utm_source=google&utm_medium=cpc&utm_campaign=_chart_es_tier2_mixed_search_brand_exact_&km_CPC_CampaignId=1501207859&km_CPC_AdGroupID=63362176052&km_CPC_Keyword=lucidchart&km_CPC_MatchType=e&km_CPC_ExtensionID=&km_CPC_Network=g&km_CPC_AdPosition=&km_CPC_Creative=294077812209&km_CPC_TargetID=aud-921551090622:kwd-

33511936169&km_CPC_Country=9069516&km_CPC_Device=c&km_CPC_placement=&km_CPC_target=&gad_source=1&gad_campaignid=1501207859&gclid=Cj0KCQiAm9fLBhCQARIsAJoNOcvIj-DNrwr8D500skx0qbTwFfI8QTFc-tKAwnPa6OlAtT6XUOMI50YaAmEhEALw_wcB

Python 3.14 documentation. (s. f.). Python Documentation. https://docs.python.org/3/

Camacho-Collados, J., & Pilehvar, M. T. (2018). From Word To Sense Embeddings: A Survey on Vector Representations of Meaning. *Journal Of Artificial Intelligence Research*, *63*, 743-788. https://doi.org/10.1613/jair.1.11259

Jeong, C. (2023). Generative AI service implementation using LLM application architecture: based on RAG model and LangChain framework. *koreascience.kr*. https://doi.org/10.13088/jiis.2023.29.4.129

Chat with data. (2023, 27 marzo). *GPT-4 Tutorial: How to Chat With Multiple PDF Files (~1000 pages of Tesla's 10-K Annual Reports)* [Vídeo]. YouTube. https://www.youtube.com/watch?v=Ix9WIZpArm0