Abstract  
Índice de figuras  
Glosario  
AGE: Administración General del Estado.  
API (Application Programming Interfaces): Interfaz de programación de aplicaciones. Representa una capacidad de comunicación entre componentes de software. Son un conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar aplicaciones software.  
AAPP: Administraciones públicas  
Interoperabilidad: Capacidad de los sistemas de información de compartir datos y conocimiento entre ellos.  
LinkedData: Método de publicación de datos estructurados con la finalidad de mejorar la interoperabilidad de los sistemas. Se basa en tecnologías web estándar.  
LMS (Learning Management Systems): Sistema de gestión de aprendizaje online.  
URL (Uniform Resource Locator): Conjunto de caracteres que sigue un estándar e identifica y permite localizar un recurso en internet.

# Capítulo 1. Introducción

Capítulo 1. Introducción

Contexto y motivación

Digitalización en las administraciones públicas

La transformación digital de las administraciones públicas españolas se ha acelerado significativamente con la irrupción de la inteligencia artificial generativa. Las Leyes 39/2015 y 40/2015 establecieron los fundamentos de la administración electrónica, pero la Estrategia Nacional de Inteligencia Artificial, aprobada en mayo de 2024, "refuerza el papel de la IA como motor de transformación tecnológica, innovación” . Planes autonómicos como GEN Digital 2025 incorporan específicamente estas tecnologías para modernizar la relación administración-ciudadanía .

La IA generativa "está destinada a transformar la economía" y las administraciones públicas no son ajenas a esta revolución. Sin embargo, el marco normativo se ha desarrollado paralelamente: desde agosto de 2025, "comienzan a aplicarse las normas de la UE sobre modelos de IA de uso general, lo que aporta más transparencia, seguridad y rendición de cuentas" . En España, las guías CCN-TEC 014, publicadas por el Centro Criptologico Nacional (CCN) establecen requisitos específicos de seguridad para sistemas de IA en administraciones públicas .

Esta convergencia entre digitalización tradicional e IA generativa crea nuevas oportunidades pero también nuevos desafíos. La fragmentación de sistemas existente se agrava por la necesidad de integrar capacidades conversacionales avanzadas manteniendo cumplimiento normativo, trazabilidad y seguridad, aspectos críticos regulados por el AI Act europeo que "regula la IA de propósito general" desde 2025

Retos de acceso a información institucional

La creciente digitalización administrativa ha generado un ecosistema informativo cada vez más disperso y complejo, compuesto por normativas, resoluciones, actas, bases de datos relacionales, expedientes electrónicos, hojas de cálculo, documentos PDF, aplicaciones web, APIs y plataformas externas. Esta heterogeneidad de fuentes y formatos plantea un importante reto para el acceso eficiente al conocimiento institucional.

A pesar de los esfuerzos por modernizar la gestión documental y avanzar en la administración electrónica, los técnicos municipales siguen enfrentándose a largos tiempos de búsqueda de información, dependencia de herramientas especializadas y una elevada carga cognitiva. Este escenario no solo dificulta la toma ágil de decisiones, sino que también limita la incorporación de nuevo personal y ralentiza los procesos de atención ciudadana.

Ante esta problemática, los avances en inteligencia artificial —especialmente los modelos de lenguaje de gran tamaño (LLM, por sus siglas en inglés)— ofrecen nuevas oportunidades para construir sistemas conversacionales capaces de simplificar el acceso a la información. En particular, la arquitectura Retrieval-Augmented Generation (RAG) permite combinar la capacidad generativa de los LLM con mecanismos de recuperación semántica de contexto relevante, favoreciendo respuestas más precisas, trazables y contextualizadas.

1.2. Problemática

La aplicación de estas tecnologías en contextos públicos sensibles exige abordar retos adicionales. Entre ellos destacan la necesidad de garantizar la privacidad de los datos, el cumplimiento de las directrices de ciberseguridad establecidas por organismos como el CCN, y la capacidad de operar con independencia de proveedores externos mediante modelos locales.

Además, las administraciones requieren soluciones modulares, escalables y adaptables, que permitan integrar de forma segura múltiples fuentes de conocimiento, preservar la soberanía del dato y enriquecer las respuestas mediante representaciones semánticas estructuradas, como los grafos de conocimiento.

1.2.1. Dispersión de fuentes de conocimiento

Párrafo 1: La información administrativa municipal se encuentra fragmentada en múltiples repositorios digitales sin conectividad entre ellos. Un expediente urbanístico típico puede requerir consultas al catastro municipal, la base de datos del Plan General de Ordenación Urbana, el registro de licencias históricas, las actas del pleno municipal donde se aprobó la normativa aplicable, y documentos PDF almacenados en carpetas compartidas departamentales.

Párrafo 2: Esta dispersión genera ineficiencias operativas significativas. Los técnicos municipales dedican entre 30-40% de su tiempo laboral a localizar información relevante antes de poder abordar el análisis del problema planteado. La ausencia de indexación semántica departamental obliga a realizar búsquedas manuales en sistemas inconexos, cada uno con interfaces y metodologías de consulta diferentes.

Párrafo 3: La heterogeneidad de formatos agrava la problemática: desde bases de datos relacionales hasta documentos escaneados, pasando por hojas de cálculo departamentales y páginas web desactualizadas. Esta diversidad técnica no solo complica la búsqueda, sino que introduce inconsistencias en la versionalidad y fiabilidad de la información recuperada.

1.2.2. Limitaciones de sistemas tradicionales

Párrafo 1: Los motores de búsqueda convencionales implementados en la mayoría de administraciones locales operan exclusivamente mediante coincidencias lexicales exactas. Cuando un ciudadano pregunta sobre "construcción de una piscina en parcela rústica", el sistema no logra relacionar conceptos semánticamente vinculados como "obra menor", "suelo no urbanizable" o "autorizaciones sectoriales", perdiendo información contextual crucial.

Párrafo 2: La gestión temporal de la información representa otro obstáculo crítico. Las normativas derogadas permanecen accesibles sin marcado temporal claro, generando confusión sobre su aplicabilidad actual. Los sistemas tradicionales carecen de capacidades de versionado inteligente que permitan distinguir automáticamente entre regulación vigente y disposiciones históricas, obligando a verificaciones manuales costosas en tiempo.

Párrafo 3: La curva de aprendizaje para empleados públicos noveles se prolonga meses debido a la complejidad de navegación entre sistemas dispares. El conocimiento institucional queda frecuentemente vinculado a personas específicas que han desarrollado expertise particular, creando dependencias críticas y riesgo de pérdida de conocimiento organizacional durante procesos de jubilación o rotación de personal.

1.2.3 Control sobre las fuentes

Párrafo 1: La verificación de autenticidad documental presenta desafíos particulares en entornos administrativos donde la integridad de la información resulta legalmente vinculante. Los sistemas tradicionales carecen de mecanismos robustos de trazabilidad que permitan identificar el origen exacto, fecha de actualización y responsable de cada fragmento informativo utilizado en la generación de respuestas.

Párrafo 2: El cumplimiento del Esquema Nacional de Seguridad exige garantías específicas sobre confidencialidad, integridad y disponibilidad de datos que los sistemas de IA externa no pueden asegurar completamente. La arquitectura RAG permite mantener el control total sobre las fuentes de conocimiento, implementando capas de seguridad personalizadas y auditorías de acceso que satisfacen los requisitos normativos del sector público.

Párrafo 3: La soberanía del dato administrativo requiere capacidades de actualización controlada y versionado granular que permitan reflejar cambios normativos de forma inmediata y trazable. Un sistema RAG bien diseñado facilita la incorporación de nuevas disposiciones legales manteniendo el historial de modificaciones, algo fundamental para la transparencia administrativa y la rendición de cuentas que demandan los ciudadanos.

1.3. Objetivos

El presente Trabajo Final de Máster tiene como objetivo principal el diseño, desarrollo y evaluación de un chatbot conversacional interno para administraciones locales, orientado al uso exclusivo por parte del personal técnico municipal. La propuesta incorpora las siguientes metas específicas:

Objetivos específicos ordenados por secuencia de diseño e implementación:

Arquitectura y Fundamentos

Diseñar una arquitectura segura y modular basada en tecnologías abiertas

Cumplir los requisitos de seguridad establecidos en CCN-TEC 014

Sistema de Ingesta y Procesamiento

Implementar un sistema RAG que integre fuentes heterogéneas (documentos, APIs, URLs, BBDD)

Embeddings y Representación Vectorial

Evaluar all-MiniLM-L6-v2 para embeddings multilingües optimizados

Comparar con intfloat/multilingual-e5-base para contextos administrativos españoles

Medir impacto en calidad de recuperación semántica según tipo de documento municipal

Vector Stores y Almacenamiento

Implementar y evaluar FAISS como solución de alta eficiencia para búsqueda de similitudes

Desarrollar implementación paralela con ChromaDB como base de datos vectorial local

Comparar métricas de rendimiento: velocidad, latencia, memoria y escalabilidad

Modelos de Lenguaje y Evaluación

Comparar el rendimiento entre modelos de OpenAI API y múltiples modelos locales

Evaluar diferentes configuraciones de parámetros en los mismos modelos

Extensiones Semánticas

Explorar grafos de conocimiento para enriquecimiento semántico

# Capítulo 2. Marco teórico

Capítulo 2. Marco teórico

Los modelos de lenguaje han experimentado en los últimos años un gran auge en las administraciones públicas, en concreto su uso se ha extendido para automatizar el acceso a la información. En el presente capítulo se analizan las principales vías líneas de investigación relacionadas con los sistemas conversacionales basados en inteligencia artificial. Se presta especial atención a las arquitecturas Retrieval-Augmented Generation (RAG), los distintos modelos de lenguaje grandes (LLM) tanto en entornos locales como en la nube y el uso de los grafos de conocimiento como mecanismo de enriquecimiento semántico.

A lo largo de este capítulo se examinan soluciones implantadas en entornos reales, así como las distintas herramientas open-source disponibles. También se realiza un análisis de la seguridad de estos entornos para su uso en las administraciones públicas a partir de las guías de seguridad de Centro Criptológico Nacional (CCN) y se realiza una revisión sobre la literatura científica con en fin de establecer una base teórica sólida en la implementación del prototipo propuesto en el trabajo.

2.1 Chatbots en la administración pública

En los últimos años la adopción de chatbots en la administración pública está experimentando un enorme crecimiento. Si bien es cierto que se llevan utilizando desde hace una década, la aparición de los LLM ha supuesto un auge en el uso de estos en todos los ámbitos de la sociedad, y en concreto han sido adoptados con gran entusiasmo por parte de las administraciones públicas.

Las principales estrategias europeas, nacionales y autonómicas en materia de inteligencia artificial y transformación digital han alineado algunas de sus directrices en la adopción de la chatbots basados inteligencia artificial por parte de las administraciones públicas. A nivel europeo, el Plan Coordinado de Inteligencia Artificial publicado en el año 2018 y actualizado en el año 2021 tiene como objetivo aplicar estrategias y programas de IA y armonizar la política de IA para evitar la fragmentación en Europa . Dentro de este marco estratégico se promueve el uso de los sistemas de IA en los servicios públicos siempre que se respeten los principios de transparencia, inclusividad, seguridad, sostenibilidad y sean respetuosos con los derechos fundamentales de las personas.

También en el ámbito europeo la reciente aprobación del Reglamento (UE) 2024/1689 (también conocido como la Ley de la Inteligencia Artificial (RIA)), indica que los el uso de los chatbots administrativos pueden ser categorizados como sistemas de IA de alto riesgo en el caso de afecten a derechos fundamentales o acceso a servicios esenciales. En estos casos deberán cumplir unos requisitos estrictos entre los que se encuentran una documentación técnica completa, la gestión de riesgos, transparencia y supervisión humana entre otros.

A nivel nacional, Estrategia de Inteligencia Artificial 2024 (ENIA) publicada por la Secretaríaa de Estado de Digitalización e Inteligencia Artificial, incluye el uso de los chatbots y asistentes virtuales en el ámbito del sector público como prioritario por su capacidad de interpretar el lenguaje natural, resolver consultas frecuentes y facilitar los trámites automatizados . Dentro de esta estrategia pretende avanzar hacia una administración proactiva y predictiva anticipándose a las necesidades ciudadanas y la optimización de los recursos públicos. En la misma línea que las estrategias europeas enmarca el uso de la IA en las AAPP dentro de una gobernanza responsable que garantice la transparencia, trazabilidad de los algoritmos y la no discriminación del acceso a los servicios públicos planteándola no solo como un vector de modernización tecnológica, sino como una palanca para mejorar la confianza, la equidad y la eficacia del Estado ante la ciudadanía.

También el Plan de Digitalización de las Administraciones Públicas 2021-2025 , impulsado por el Ministerio de Asuntos Económicos y Transformación Digital además de destacar el potencial de la implantación de la IA en las administraciones públicas españolas, establece como uno de los pilares claves del plan el uso de soluciones inteligentes como los asistentes virtuales para la mejora de la eficiencia administrativa y la experiencia ciudadana, centrada en el dato, interoperables y proactiva

A nivel autonómico, la Comunitat Valenciana impulsa el desarrollo de soluciones conversacionales como parte del Plan Estratégico de Transformación Digital de la Administración de la Generalitat, GEN Digital 2025 , el cual promueve el uso de tecnologías como los chatbots para facilitar la relación entre ciudadanía y administración desde una perspectiva accesible, inclusiva y eficiente. Además, el plan impulsa un modelo de gobernanza del dato a través de proyecto Ànima para coordinar un despliegue seguro, ético e interoperable de las soluciones tecnológicas de toda la administración valenciana.

En cuanto a nivel local, numerosas ciudades están implantando marcos regulatorios del uso de la IA en sus administraciones. El Ayuntamiento de Barcelona a través de su medida de gobierno “Intel·ligència Artificial per a una Administració Ètica i al Servei de les Persones” establece un marco ético y normativo para el uso de la IA a municipal, basado en los principios de transparencia, explicabilidad, supervisión humana, equidad y responsabilidad.Por su parte, el Ayuntamiento de Madrid a través del “Itinerario para el uso de la Inteligencia Artificial en el Ayuntamiento de Madrid.” desarrolla una estrategia propia en consonancia con el Ayuntamiento de Barcelona. Ambas iniciativas evidencian la necesidad de adoptar los chatbots desde una perspectiva normativa, funcional y participativa.

Cade vez más administraciones están implementando soluciones basadas en inteligencia artificial, en muchos casos en forma de chatbots. Sin embargo, en muchos casos es necesario desarrollar un marco normativo propio claro. Estas políticas deben garantizar el cumplimiento de los principios fundamentales recogidos en el RIA, como la transparencia, la trazabilidad, la supervisión humana, la seguridad, la no discriminación y el respeto a los derechos fundamentales, promoviendo así una transformación digital confiable y centrada en las personas en el ámbito municipal. Los asistentes conversacionales en la administración pública han emergido como herramientas clave para mejorar la eficiencia y la accesibilidad.

En este contexto, resulta conveniente realizar un análisis de experiencias reales de la adopción de chatbots. por parte de las administraciones públicas Así pues en un estudio reciente Larsen y Følstad (2024) ponen en valor los chatbots conversacionales municipales como el chatbot noruego Kommune-Kari, mejora el acceso de la ciudadanía a la información simplificando procesos administrativos y aportando información sobre las necesidades de la ciudadanía En el plano español existen numerosas experiencias como las recopiladas por el Instituto Nacional de las Administraciones Públicas(INAP) y el portal de datos abiertos del Gobierno de España datos.gob.es donde se demuestra que el uso de los chatbots mejora la transparencia y democratiza el acceso a la información pública a la ciudadanía.

También es necesario destacar el informe del Joint Research Centre (JEC) europeo sobre interoperabilidad de chatbots donde se subraya la importancia de estandarizar esta tecnología para garantizar su integración entre plataformas, mejorar la experiencia ciudadana y reforzar la confianza en el canal digital.

En este contexto, resulta conveniente realizar un análisis de experiencias reales de la adopción de chatbots por parte de las administraciones públicas. Así pues, en un estudio reciente Grøndahl y Følstad (2024) ponen en valor los chatbots conversacionales municipales como el chatbot noruego Kommune-Kari, que mejora el acceso de la ciudadanía a la información, simplificando procesos administrativos y aportando información sobre las necesidades de la ciudadanía [10]. En el plano español existen numerosas experiencias como las recopiladas por el Instituto Nacional de las Administraciones Públicas (INAP) [11] y el portal de datos abiertos del Gobierno de España, datos.gob.es [12], donde se demuestra que el uso de los chatbots mejora la transparencia y democratiza el acceso a la información pública por parte de la ciudadanía.

También es necesario destacar el informe del Joint Research Centre (JRC) europeo sobre interoperabilidad de chatbots [13], donde se subraya la importancia de estandarizar esta tecnología para garantizar su integración entre plataformas, mejorar la experiencia ciudadana y reforzar la confianza en el canal digital.

Otros informes recientes complementan y amplían esta evidencia. Por ejemplo, el estudio "Bitácora de la IA en Europa" de ITGespub recopila más de 70 casos de uso de chatbots en administraciones europeas, desde el chatbot ADA de la administración belga hasta el sistema IRIS en Finlandia, reflejando una tendencia generalizada hacia la automatización de servicios y atención ciudadana. Por su parte, la base de datos de casos de IA en el sector público del Joint Research Centre identifica también experiencias con chatbots en contextos regionales, nacionales y municipales, proporcionando ejemplos verificables, con categorías como salud, empleo, justicia o educación.

Ilustración 1. Overview of the main themes of our analysis concerning the impact of chatbots structured according to the public value dimensions of the (2019) framework.

En el caso del estudio de Larsen y Følstad [10], los resultados se presentan también mediante gráficos cualitativos que muestran las ventajas y barreras percibidas por la ciudadanía al utilizar asistentes conversacionales. Uno de los gráficos más destacados revela que el 73% de los participantes valora positivamente la inmediatez de respuesta de los chatbots, mientras que el 61% expresa preocupación por la falta de contacto humano. Esta información empírica permite reforzar la necesidad de establecer un equilibrio entre eficiencia tecnológica y garantías institucionales.

En conjunto, estos informes y estudios avalan el desarrollo de este trabajo de fin de máster al demostrar la creciente relevancia, impacto y proyección de los chatbots en el sector público. Aportan evidencia empírica, marcos normativos y estrategias institucionales que legitiman la necesidad de investigar, evaluar e implementar este tipo de soluciones conversacionales bajo criterios de eficacia, responsabilidad y servicio a la ciudadanía.

Estos hallazgos justifican los pilares del presente TFM: una arquitectura RAG para enriquecer las respuestas, control y trazabilidad de la base de conocimiento, y mecanismos de transparencia y explicabilidad.

Elementos clave de diseño y justificación del TFM

Este trabajo se sustenta sobre tres pilares fundamentales que emergen con claridad de los estudios analizados:

1. Uso de una arquitectura basada en RAG (Retrieval-Augmented Generation)

El uso de RAG permite enriquecer las respuestas del modelo con información precisa extraída desde fuentes internas reales (documentos municipales, APIs, bases de datos), lo que contribuye a mejorar significativamente la contextualización, relevancia y trazabilidad de las respuestas. Esta arquitectura responde directamente al problema de literalidad y falta de profundidad semántica identificado en estudios recientes .

2. Control sobre la base de conocimiento y soberanía tecnológica

Uno de los factores críticos de éxito en los proyectos de chatbots en administraciones públicas es el mantenimiento de la base de conocimiento y el control sobre las fuentes que alimentan al sistema . El presente trabajo aborda esta cuestión mediante una arquitectura modular que separa y gestiona las fuentes de forma estructurada, incluyendo mecanismos de actualización automática y trazabilidad por origen. Esta aproximación favorece la soberanía tecnológica, la privacidad de los datos y el cumplimiento normativo.

3. Transparencia y definición de límites del sistema

La falta de transparencia sobre el funcionamiento y limitaciones del sistema genera desconfianza entre los usuarios. Por ello, el prototipo desarrollado incorpora mecanismos para mostrar qué fuente ha sido usada para generar una respuesta, así como advertencias explícitas sobre los límites del sistema. Esta funcionalidad se apoya tanto en el diseño de interfaz como en la lógica de trazabilidad del sistema RAG.

A continuación, se resumen estos elementos clave en la siguiente tabla:

Conclusiones

El presente TFM responde a retos comunes detectados en la literatura académica sobre chatbots públicos. La arquitectura propuesta incorpora mecanismos de recuperación aumentada (RAG), control sobre las fuentes y transparencia, alineándose con recomendaciones de estudios internacionales y nacionales. Este enfoque permite avanzar hacia sistemas más útiles, confiables y adaptados al entorno de las administraciones locales.

El uso de RAG mejora la contextualización y trazabilidad de las respuestas.

El control sobre las fuentes garantiza soberanía tecnológica y cumplimiento normativo.

La transparencia del sistema fortalece la confianza de los usuarios.

Estos elementos convierten al prototipo en una solución escalable y replicable.

2.2. Modelos de lenguaje: OpenAI vs modelos locales

Los modelos de lenguaje de gran tamaño (Large Language Models, LLMs) han evolucionado rápidamente en los últimos años, consolidándose como la tecnología base para múltiples aplicaciones de generación de lenguaje natural. Actualmente, los modelos más destacados incluyen:

GPT-4 (OpenAI, 2023): modelo de propósito general con capacidades avanzadas de razonamiento, multilingüismo y codificación. Disponible a través de la API de OpenAI y Azure OpenAI. Es el modelo base del servicio ChatGPT Plus y permite integrar conocimiento externo mediante RAG

LLaMA 2 y LLaMA 3 (Meta, 2023–2024): modelos abiertos de código disponibles en múltiples tamaños (7B, 13B, 70B), con un enfoque en eficiencia y rendimiento competitivo. LLaMA 3 mejora el alineamiento y ha sido entrenado con un corpus más extenso

Mistral (Mistral AI, 2023): modelos ligeros y eficientes (7B) diseñados para uso local, con variantes como Mixtral (modelo Mixture of Experts de 12,9B parámetros) que mejora el rendimiento sin aumentar drásticamente el coste computacional

Phi (Microsoft, 2024): modelos optimizados para tareas educativas y uso controlado, como Phi-2 (2,7B), con enfoque en eficiencia y generación de lenguaje alineada .

Gemma (Google, 2024): nuevo modelo open-weight orientado a facilitar la creación de asistentes IA personalizados, con licencia Apache 2.0 y compatibilidad directa con frameworks como Ollama o HuggingFace .

Desde una perspectiva arquitectónica, pueden distinguirse dos enfoques principales para su implementación en sistemas de atención en la administración pública: el uso de modelos ofrecidos como servicio a través de APIs externas (como los de OpenAI), y la ejecución local de modelos en servidores propios, empleando soluciones como Ollama o LM Studio.

Los modelos basados en API, como los de OpenAI, ofrecen múltiples ventajas: acceso inmediato a modelos punteros, actualizaciones frecuentes, escalabilidad bajo demanda y facilidad de integración. Sin embargo, su uso conlleva importantes consideraciones en términos de privacidad, dependencia tecnológica, trazabilidad y costes acumulativos, lo que puede ser problemático en entornos administrativos con altos requisitos de protección del dato

Por otro lado, la ejecución de modelos localmente permite una mayor soberanía tecnológica y control del procesamiento de información. Plataformas como Ollama facilitan el despliegue de modelos abiertos como LLaMA, Mistral o TinyLLaMA en entornos seguros, sin necesidad de exponer datos a servicios externos. Aunque esta opción requiere mayor capacidad computacional y experiencia técnica, permite cumplir con los principios de confidencialidad y auditabilidad definidos en marcos normativos como el Esquema Nacional de Seguridad y la guía CCN-TEC 014 .

Adicionalmente, la posibilidad de ajustar parámetros internos de los modelos locales (como temperatura, top-k o top-p) abre la puerta a experimentos controlados para analizar cómo afectan estas configuraciones al comportamiento del sistema. Esta capacidad de ajuste fino no siempre está disponible o es limitada en servicios en la nube, lo que refuerza el valor del enfoque local para tareas de evaluación comparativa.

En este TFM, se plantea una comparativa entre ambos enfoques: modelos accesibles mediante API (OpenAI) frente a modelos ejecutados en local (Ollama). Esta comparación no solo se realizará entre distintos modelos, sino también entre distintas configuraciones de un mismo modelo, con el objetivo de estudiar su impacto en métricas como la precisión, la latencia, la relevancia de la respuesta o el coste computacional.

Este enfoque permite avanzar en la evaluación de sistemas de IA generativa aplicados al ámbito público, aportando evidencia práctica sobre su rendimiento, sus requisitos técnicos y sus implicaciones en términos de gobernanza tecnológica.

2.3. Recuperación aumentada de información (RAG)

Uno de los principales retos en la aplicación práctica de modelos generativos en contextos institucionales es la actualización, personalización y trazabilidad del conocimiento. Para resolver este problema, en los últimos años ha cobrado gran relevancia la arquitectura conocida como Retrieval-Augmented Generation (RAG), que combina técnicas de recuperación semántica con generación controlada por contexto.

En un sistema RAG, el modelo de lenguaje no responde únicamente a partir de su conocimiento interno, sino que accede dinámicamente a un corpus documental vectorizado. Esto permite obtener fragmentos relevantes a partir de bases de datos, documentos institucionales, normativas o incluso APIs externas, y utilizarlos como contexto explícito en la generación de la respuesta final

El proceso RAG consta habitualmente de tres etapas principales:

Figura 1. R EPRESENTACIÓN ESQUEMÁTICA DEL PROCESO DE RECUPERACIÓN AUMENTADA DE INFORMACIÓN (RAG), INCLUYENDO ETAPAS DE VECTORIZACIÓN, RECUPERACIÓN Y GENERACIÓN. ELABORACIÓN PROPIA BASADA EN , Y

Recuperación semántica: Ante una consulta, se recuperan los fragmentos más relevantes mediante similitud vectorial, aplicando técnicas como HNSW o Reranking

Generación condicionada: El LLM genera una respuesta usando tanto la consulta como los fragmentos recuperados, lo que mejora la precisión, trazabilidad y transparencia del sistema.

Generación condicionada: El LLM genera una respuesta usando tanto la consulta como los fragmentos recuperados, lo que mejora la precisión, trazabilidad y transparencia del sistema

La principal ventaja de esta arquitectura es que permite actualizar el conocimiento sin reentrenar el modelo, controlar las fuentes utilizadas y generar respuestas respaldadas en documentos accesibles, reforzando así la confianza del usuario y el cumplimiento normativo. Estas cualidades son especialmente valiosas en el contexto de las administraciones públicas, donde la trazabilidad, la fiabilidad y la transparencia son fundamentales.

Este enfoque, además de alinearse con las buenas prácticas internacionales, responde directamente a los problemas identificados en el apartado 2.1 sobre literalidad, falta de actualización y dificultad de trazabilidad en los sistemas de chatbot tradicionales.

2.4. Integración de fuentes heterogéneas en sistemas RAG

Uno de los aspectos más desafiantes en la construcción de sistemas RAG robustos para administraciones públicas es la necesidad de integrar fuentes de información heterogéneas. A diferencia de sistemas cerrados o entrenados únicamente con datos genéricos, las aplicaciones institucionales requieren responder a partir de normativas, bases de datos, APIs municipales, documentos ofimáticos y páginas web oficiales.

Este enfoque multifuente exige técnicas de ingesta, limpieza, transformación y enriquecimiento que permitan armonizar los datos estructurados y no estructurados en un mismo índice semántico. En la literatura reciente se han propuesto arquitecturas que distinguen los canales de ingestión por tipo de fuente, permitiendo aplicar estrategias específicas a cada una:

Documentos: requieren procesamiento OCR (en caso de PDF escaneado), segmentación por secciones, extracción de metadatos y almacenamiento como fragmentos vectorizados .

APIs: requieren autenticación, programación de consultas periódicas (cron) y normalización de los datos a un formato indexable (JSON simplificado, YAML, etc.) .

Bases de datos: necesitan integración vía conectores SQL o NoSQL, selección de campos clave y anonimización si contienen información sensible

Webs: requieren crawling o scraping controlado, limpieza de HTML y seguimiento de enlaces con lógica de profundidad .

La integración de estas fuentes es esencial para asegurar que el sistema proporcione respuestas trazables, actualizadas y relevantes. Para ello, en este TFM se propone una arquitectura modular de ingestión basada en pipelines independientes por tipo de fuente, todos convergiendo en un índice común (FAISS) enriquecido con metadatos de origen.

Este diseño garantiza:

Independencia de actualización por fuente.

Registro de procedencia y trazabilidad.

Aplicación de filtros semánticos o de control de calidad por canal.

Mayor cobertura de información institucional y respuesta contextualizada.

Así, la integración heterogénea no solo es un requisito técnico, sino un pilar de transparencia, eficiencia y resiliencia del sistema conversacional.

En el contexto del presente trabajo, resulta esencial comparar las características, ventajas y limitaciones de los modelos de lenguaje ejecutados localmente frente a los ofrecidos como servicio a través de APIs comerciales. Esta comparativa orienta las decisiones arquitectónicas y metodológicas del sistema propuesto.

La siguiente tabla resume las diferencias clave entre ambas aproximaciones:

Desde el punto de vista de una administración pública, los modelos locales ofrecen ventajas en términos de cumplimiento normativo (ENS, CCN-TEC 014), protección de datos y personalización. No obstante, requieren una mayor inversión inicial y capacidades técnicas.

Por su parte, los modelos comerciales permiten una integración rápida y eficiente, ideal para entornos sin recursos propios, pero a costa de sacrificar control, trazabilidad y soberanía tecnológica.

Este trabajo opta por integrar ambos enfoques de forma comparativa, permitiendo así evaluar empíricamente su rendimiento, costes y adecuación a un entorno público local.

2.5. Seguridad y cumplimiento normativo: Guía CCN-TEC 014

La seguridad es un pilar esencial en el despliegue de sistemas de inteligencia artificial dentro de las administraciones públicas. En el contexto español, la Guía CCN-TEC 014 del Centro Criptológico Nacional (CCN-CERT) establece una serie de recomendaciones específicas para la protección de sistemas que incorporan capacidades de IA, en línea con el Esquema Nacional de Seguridad (ENS).

Esta guía aborda los riesgos inherentes al uso de IA, como la opacidad del procesamiento, la manipulación de modelos, la generación de respuestas incorrectas o no trazables y la posible fuga de información. A partir de ello, establece requisitos en las siguientes áreas clave:

Control de acceso: los sistemas deben garantizar autenticación robusta, separación de roles y registro de actividad.

Auditabilidad y trazabilidad: toda respuesta generada por un modelo debe estar vinculada a un historial de consulta y a su fuente de información.

Supervisión humana: los sistemas deben permitir la revisión y validación de respuestas automáticas, especialmente en contextos críticos.

Protección del modelo: se recomienda cifrado del modelo, control de integridad y actualizaciones controladas.

Privacidad de los datos: la información sensible debe ser anonimizada, y el sistema debe impedir el entrenamiento no deseado con datos reales.

Gobernanza del sistema: debe existir una política clara de gestión de datos, entrenamiento, evaluación de resultados y respuesta ante incidentes.

En el caso del presente TFM, estas recomendaciones se traducen en una arquitectura que:

Separa los módulos de recuperación y generación para garantizar trazabilidad.

Almacena un log de consultas y respuestas asociado a cada sesión.

Permite al usuario visualizar la fuente documental usada para generar la respuesta.

No entrena los modelos con datos introducidos por los usuarios.

Asegura que todos los datos sensibles se mantengan en entornos controlados.

El cumplimiento de estas directrices no solo fortalece la seguridad del sistema, sino que además refuerza la confianza institucional y el alineamiento con la legislación vigente en materia de protección de datos (RGPD), ENS y directivas europeas sobre uso responsable de la inteligencia artificial.

2.6. Grafos de conocimiento y recuperación semántica

Los grafos de conocimiento representan una estructura formalizada para modelar entidades, relaciones y atributos en un dominio determinado. Su integración en sistemas de inteligencia artificial ha demostrado ser especialmente útil para enriquecer procesos de recuperación semántica, razonamiento simbólico y generación de respuestas contextualizadas.

A diferencia de una base de datos tradicional o un conjunto de documentos vectorizados, un grafo de conocimiento permite representar el significado de forma estructurada mediante tripletas del tipo (sujeto, predicado, objeto). Esta capacidad es fundamental para establecer conexiones explícitas entre conceptos, inferir nueva información y validar la coherencia de las respuestas.

En el ámbito de la administración pública, los grafos de conocimiento pueden modelar:

Catálogos normativos y sus relaciones jerárquicas.

Entidades organizativas y competencias administrativas.

Procedimientos, plazos, documentos requeridos y responsables.

Relaciones entre proyectos, financiación y convocatorias.

Desde el punto de vista técnico, los grafos pueden implementarse mediante soluciones como RDF (Resource Description Framework), OWL (Web Ontology Language) o almacenes de grafos como Neo4j o GraphDB. Su combinación con sistemas de lenguaje natural puede seguir dos aproximaciones principales:

Inyección de contexto estructurado: los nodos y relaciones relevantes al prompt se convierten en texto natural y se incorporan a la consulta del modelo.

Razonamiento externo: se consulta el grafo como paso previo, extrayendo hechos o relaciones que luego se validan o complementan mediante el LLM.

En este TFM, se contempla la integración futura de un grafo de conocimiento que represente la estructura institucional, los procedimientos y las fuentes normativas del Ayuntamiento. Este grafo servirá para:

Validar respuestas generadas por el modelo.

Razonar sobre dependencias entre entidades o fases de tramitación.

Mejorar la interpretación semántica de consultas complejas.

Facilitar tareas de auditoría y trazabilidad del conocimiento utilizado.

Su incorporación representa un paso hacia sistemas conversacionales más semánticos, explicables y alineados con el conocimiento institucional real.

# Capítulo 3. Metodología

Capítulo 3. Metodología

3.1. Diseño general del sistema

El sistema desarrollado sigue una arquitectura modular que responde a los principios de recuperación aumentada (RAG), comparabilidad entre modelos y trazabilidad de respuestas, siendo estos aspectos ampliamente destacados como críticos en la literatura científica reciente sobre chatbots aplicados a la administración pública [4][5][6].

El objetivo principal del diseño es proporcionar una plataforma conversacional robusta, segura y adaptada al conocimiento institucional, que permita a técnicos municipales consultar información mediante lenguaje natural y recibir respuestas relevantes, documentadas y trazables. Para ello, se han articulado distintos módulos funcionales que se interconectan, respetando los principios de separación de responsabilidades, escalabilidad y gobernanza del conocimiento [7][8].

3.1.1 Arquitectura modular basade en recuperación aumentada

La arquitectura propuesta se basa en una separación clara de responsabilidades entre los siguientes bloques funcionales:

Ingesta y vectorización de datos heterogéneos. La ingesta de conocimiento se realiza a partir de múltiples fuentes municipales: documentos normativos, PDFs escaneados, bases de datos institucionales, APIs y páginas web. Cada tipo de fuente se trata mediante un pipeline específico. El contenido se procesa, se divide en fragmentos y se vectoriza con modelos de sentence-transformers, técnica ampliamente utilizada para tareas de recuperación semántica [5][18].

Índice semántico y motor de recuperación. Se utiliza FAISS (Facebook AI Similarity Search) como motor de búsqueda vectorial para almacenar y recuperar los fragmentos más similares a cada consulta [4]. La recuperación se realiza por similitud coseno, y se limita a los k fragmentos más relevantes.

Generación de respuesta condicionada (RAG). Los fragmentos recuperados se utilizan para construir un prompt enriquecido que alimenta el modelo LLM elegido. Esta técnica, conocida como Retrieval-Augmented Generation (RAG), ha demostrado mejorar la precisión y la trazabilidad en sistemas conversacionales aplicados a dominios sensibles [16][18][19].

Comparador de modelos. El sistema permite enviar la misma pregunta y contexto a dos motores distintos (OpenAI y modelo local via Ollama), y comparar sus respuestas, latencias y tokens usados. Este enfoque comparativo está alineado con estudios recientes que proponen evaluaciones controladas entre modelos de lenguaje [8][20].

Interfaz web y capa de control. La interfaz web muestra al usuario no solo la respuesta generada, sino también los documentos o fragmentos utilizados, la fuente, el modelo responsable y advertencias si existen límites de conocimiento. Esto refuerza la transparencia, tal como recomiendan las guías de buenas prácticas en sistemas de IA en el sector público [6][15][20].

Esta arquitectura responde a los retos identificados en el Capítulo 2:

Literalidad y falta de actualización: solucionado mediante la recuperación semántica dinámica desde documentos reales, sin necesidad de reentrenar el modelo base [16][17].

Falta de trazabilidad: cada fragmento recuperado se acompaña de metadatos que indican su fuente original y fecha, permitiendo auditar la respuesta [15][18].

Dependencia tecnológica: el uso de modelos locales con Ollama y el control sobre la base de conocimiento fortalecen la soberanía tecnológica del sistema [3][11].

Seguridad normativa: la arquitectura permite cumplir el Esquema Nacional de Seguridad y las recomendaciones del CCN-TEC 014 mediante separación de módulos, registro de logs y ausencia de entrenamiento con datos sensibles [1][15].

3.1.2 Diagrama de arquitectura

3.1.3 Relación con otras arquitecturas RAG

La arquitectura propuesta sigue los principios recomendados por múltiples autores en el ámbito de los sistemas RAG modernos. Por ejemplo, Lewis et al. [18] establecen una arquitectura de tres fases: recuperación, reordenamiento y generación. El diseño aquí implementado sigue esta lógica, ampliándola con un módulo comparador de modelos y una interfaz con trazabilidad explícita. Autores como Fischer [16] y Humanloop [20] destacan que estas arquitecturas permiten aumentar la confianza en los modelos de IA al mostrar la fuente documental de cada respuesta, una estrategia aplicada explícitamente en este trabajo.

3.1.4 El diseño modular responde a varias necesidades clave:

El diseño modular responde a varias necesidades clave identificadas en la literatura y los requisitos funcionales de un sistema conversacional aplicado al sector público:

Escalabilidad: permite añadir nuevos tipos de fuentes de datos (por ejemplo, procedimientos electrónicos, boletines oficiales o datos en tiempo real) sin necesidad de rediseñar la arquitectura global. También facilita la incorporación futura de nuevos modelos de lenguaje o componentes especializados como agentes o razonadores simbólicos.

Comparabilidad: al separar el módulo de recuperación de información del módulo generador, es posible enviar exactamente el mismo contexto a diferentes modelos LLM. Esta separación es crucial para realizar comparaciones justas y controladas entre modelos locales y remotos, como GPT-4 frente a Mistral [8][20].

Seguridad y cumplimiento normativo: el aislamiento entre las fases de ingesta, almacenamiento, recuperación y generación permite aplicar el principio de mínimo privilegio y proteger los datos sensibles. Esta segmentación facilita también el cumplimiento de los principios establecidos en el Esquema Nacional de Seguridad y las recomendaciones de la guía CCN-TEC 014 [1][15].

Trazabilidad: cada respuesta generada incluye los fragmentos documentales utilizados, junto con sus metadatos (tipo de fuente, fecha, ubicación original). Esto permite validar las respuestas, auditar el sistema y mejorar la confianza del usuario [16][18].

3.1.6 Tecnologías utilizadas

A continuación, se describen las principales tecnologías empleadas en el desarrollo del prototipo:

Estas tecnologías han sido seleccionadas por su facilidad de integración, licencia libre, documentación activa y amplia comunidad, lo que garantiza la mantenibilidad y replicabilidad del sistema

# Capítulo 4. Sistema de ingesta y calidad de datos

Capítulo 4. Implementación

4.1 Visión general de la arquitectura RAG híbrida

Uno de los principales desafíos en la construcción de asistentes conversacionales para el sector público reside en la preparación y estructuración del conocimiento institucional, el cual suele encontrarse disperso en múltiples plataformas, formatos y sistemas. Este TFM aborda este reto mediante un enfoque de ingesta multicanal basado en la arquitectura modular presentada en la sección 3.1.

El objetivo de esta fase es recopilar, transformar y vectorizar datos procedentes de fuentes heterogéneas del Ayuntamiento, de forma que puedan ser integrados en el sistema RAG (Retrieval-Augmented Generation) y utilizados como contexto en las respuestas generadas por los modelos de lenguaje.

El sistema evoluciona desde un pipeline único a una arquitectura RAG híbrida en cuatro componentes, alineada con los requisitos de seguridad, trazabilidad y soberanía tecnológica:

Vector store privado (documentación interna): faqs internas, expedientes, actas, resoluciones no públicas.

Vector store público (corpus web + transparencia): normativa pública, sedes electrónicas, páginas de trámites, PDF públicos.

Base de conocimiento de entidades (enriquecimiento semántico): términos, organizaciones, procedimientos y referencias legales normalizadas.

Grafo de conocimiento (relaciones y workflows administrativos): referencias normativas, jerarquías, dependencias y pasos procedimentales.

Esta separación permite segmentar datos por clasificación de seguridad (público/interno/restringido/confidencial), aplicar políticas ENS/CCN y ejecutar consultas híbridas (vectores + entidades + grafo).

Durante la planificación de este proyecto se han identificado cuatro tipos principales de fuentes de conocimiento, de acuerdo con los contenidos reales utilizados en el desarrollo del sistema

Cada fuente requiere un pipeline de ingesta específico, ajustado a sus particularidades técnicas y semánticas [21][22].

4.1.1 Integración con la arquitectura del sistema (Cap. 3)

La capa de ingesta alimenta a los vector stores (privado y público). La capa de recuperación (retriever) expone un contrato unificado a la capa generativa (LLM), de modo que el mismo contexto recuperado se pasa a modelos distintos (API y locales) para comparativa controlada. Los metadatos de procedencia se conservan en todo el flujo, garantizando trazabilidad y auditoría.

4.2 Ingesta unificada de documentos y web (clasificación público/privado en origen)

4.2.1 Dataset, fuentes y supuestos (enfoque unificado)

Enfoque. La ingesta trata dos tipos de fuentes —documentos (FS/repositorio) y webs (HTTP/HTTPS)— mediante un único pipeline lógico. En tiempo de ingesta se asigna data\_classification ∈ {público, interno, restringido, confidencial} según el origen y las reglas de gobernanza; esta etiqueta determina el routing del chunk hacia el vector store público o privado.

Fuentes y alcance actuales

Documentos: PDF/DOCX/TXT/CSV procedentes de unidades municipales (carpetas controladas).

Web: dominios institucionales (sede electrónica, portal de transparencia y páginas municipales con normativa/servicios). Descubrimiento sitemapfirst y/o semillas definidas.

Supuestos operativos

Idempotencia por huella (SHA256) o mtime.

OCR cuando el PDF es imagen.

Conservación de metadatos canónicos por chunk (ruta/URL, fecha, sección/página, entidad/unidad, clasificación de seguridad).

Auditoría mediante artefactos de ejecución (stdout.txt, fetch\_index.json, summary.json).

4.2.2 Pipeline paso a paso (conceptos y justificación)

Detección de cambios (idempotencia)

Qué es. Comprobación previa que compara la huella (checksum) y/o la fecha de modificación del origen para decidir si un documento debe volver a procesarse.

Por qué. Evita reprocesados innecesarios, reduce coste y tiempo, y garantiza coherencia entre ejecuciones. Además favorece auditoría y reproducibilidad (si nada cambió, el resultado debe ser idéntico).  
Cómo. Cálculo de SHA256 del contenido (o política por mtime) antes de extraer. Si la firma coincide con la última ejecución registrada, se salta el documento.

Métrica asociada. Idempotency hit rate (porcentaje de documentos saltados por no cambio).

Extracción (parsers por formato)

Qué es. Obtención del texto y metadatos desde PDF, DOCX, TXT o CSV, conservando la estructura lógica cuando existe.

Por qué. Una extracción fiel es la base de un buen chunking y de embeddings de calidad; errores aquí se propagan al resto del pipeline.

Cómo. Lectores especializados por tipo (p.ej., PyMuPDF/pdfminer, python-docx, parsers CSV/TXT) y OCR cuando el PDF es imagen. Se registran incidencias por formato para trazabilidad.  
Métricas. Coverage por formato y error rate de extracción.

Normalización (limpieza y canonicidad)

Qué es. Depurar el texto: eliminar encabezados/pies, colapsar espacios, corregir artefactos de OCR, normalizar comillas y guiones, y preservar la estructura jurídica (artículos, capítulos, listas).  
Por qué. Reduce ruido, baja el número de tokens y mejora la recuperabilidad (recall) y la factualidad posterior al disminuir variaciones superficiales.

Cómo. Reglas regex y heurísticas para identificar boilerplate, expansión de abreviaturas administrativas (Ayto., Admón., Art., Ord.), unificación de listas y numeraciones.

Métricas. Metadata completeness y coherencia de chunk posterior.

Clasificación de seguridad (data governance)

Qué es. Etiquetado del contenido según público / interno / restringido / confidencial.

Por qué. Alinea el sistema con ENS/CCN, permite separar índices (privados/públicos) y aplicar filtros de recuperación y control de acceso.

Cómo. Reglas basadas en origen (ruta, unidad), etiquetas del documento y overrides manuales cuando proceda; el campo data\_classification se propaga a BD y vector store.

Métrica. % de chunks correctamente etiquetados y nº de incidencias.

Chunking semántico (segmentación con solape)

Qué es. División del texto en unidades coherentes (párrafos/epígrafes), con overlap para mantener contexto entre fragmentos.

Por qué. Equilibra contexto (útil al LLM) y recuperabilidad (recall@k). El solape reduce cortes bruscos en límites de párrafo.

Cómo. Heurística de párrafo con límites suaves y conservación de enumeraciones legales; tamaño objetivo configurable.

Métricas. Recall@k (cuando esté disponible), nearduplicate ratio y coherencia estimada.

Deduplicación (exacta y aproximada)

Qué es. Eliminación de duplicados exactos (por hash) y nearduplicates (por kshingles + Jaccard).

Por qué. Evita sesgos y ruido, ahorra almacenamiento e impide que respuestas citen repetidamente el mismo contenido.

Cómo. Índice de huellas y similitud con umbral configurable; en web, se añade canonicación de URL y detección de boilerplate repetido.

Métricas. Nearduplicate ratio y nº de chunks únicos.

Persistencia (BD + vector store)

Qué es. Escritura de chunks y metadatos (origen, tipo, clasificación, fecha, página/sección) en la base de datos y de embeddings en el vector store correspondiente.

Por qué. Garantiza trazabilidad, auditoría y consultas filtrables por procedencia/fecha/unidad.  
Cómo. Tablas Document/Chunk (BD) con run\_id y marcas temporales; colecciones separadas en FAISS/Chroma (público/privado).

Métricas. Latencia de escritura, errores y metadata completeness.

Especificidades por fuente (documentos y web)

A) Documentos (FS/repositorio controlado)

Detección de cambios: checksum/mtime por fichero; control de eliminaciones y renombres.

Extracción: PyMuPDF/pdfminer (PDF), python-docx (DOCX), parsers TXT/CSV; OCR cuando el PDF es imagen.

Limpieza/normalización: eliminación de cabeceras/pies, colapso de espacios, corrección OCR, preservación de estructura jurídica (artículos, epígrafes, listas).

Chunking: objetivo 900–1000 caracteres con overlap 10–12% para continuidad semántica.

Deduplicación: exacta (hash) y aproximada (kshingles + Jaccard).

Errores típicos: documentos corruptos, protegidos o con tablas complejas (se registran incidencias).

Artefactos: summary.json con totales por documento y run.

B) Web (HTTP/HTTPS)

Descubrimiento: sitemapfirst y/o semillas; allowed\_domains, robots-policy (strict/ignore según pruebas), force-https cuando aplica.

Límites defensivos: max\_pages, rate limiting, retry/backoff y control de iframes del mismo dominio.

Limpieza HTML municipal: eliminación de boilerplate (nav/header/aside/footer/scripts/styles), breadcrumbs y botones; expansión de abreviaturas (Ayto., Admón., Art., Ord.).

PDFs enlazados: reutilizan el pipeline de documentos para mejor calidad de texto.

Versionado por URL: URL canónica + hash de contenido para detectar cambios reales; deduplicación de plantillas repetidas.

Métricas web: coverage (descubiertas vs procesadas), chunks/page, errores HTTP, denegaciones por robots y nearduplicate ratio.

Estrategias de ingesta web y criterios de uso (evaluadas)

Para abordar la variabilidad de tipos de web en entornos administrativos, se han incorporado y evaluado tres estrategias complementarias de ingesta web. La orquestación elige dinámicamente la adecuada por dominio/URL, y registra en metadatos la estrategia efectiva usada.

1) Requests + BeautifulSoup (HTML estático rápido)

Qué. Descarga por HTTP(S) y parse del HTML sin ejecutar JavaScript.

Cuándo. Páginas con render serverside (CMS municipales clásicos), listados estáticos, enlaces a PDF.

Pros. Alta velocidad y bajo consumo; control fino de selectores; robusto frente a bloqueos básicos.

Contras. No ejecuta JS → puede perder contenido lazy o SPA.

Parámetros habituales. timeout, rate\_per\_host, allowed\_domains, depth, exclude (regex de extensiones), user\_agent.

2) Sitemap (descubrimiento dirigido y cobertura)

Qué. Descubre URLs desde robots.txt/sitemap.xml, priorizando por lastmod cuando existe.

Cuándo. Portales que mantienen sitemaps; útil para profundidad y capturar PDFs enlazados.

Pros. Cobertura determinista y menos ruido; evita loops de navegación.

Contras. Sitemaps a veces incompletos o desactualizados; requiere fallback de crawling.

Parámetros. robots-policy (recomendado strict), max\_pages por tipo, include/exclude patterns.

3) Selenium (render dinámico y fidelidad de DOM)

Qué. Navegador headless que ejecuta JavaScript y extrae el DOM postrender.

Cuándo. Sedes SPA, tablas paginadas con JS, contenidos lazy, iframes del mismo dominio.

Pros. Máxima fidelidad de contenido; acceso a elementos generados por JS.

Contras. Mayor latencia y coste; precisa waits explícitos; más sensible a ratelimits o captchas.

Parámetros. page\_load\_timeout, explicit waits por selectores, modo headless, scroll incremental, screenshot on failure.

Evaluación y política de selección

Métricas usadas. Coverage (descubiertas→procesadas), chunks/page, tasa de errores HTTP/JS, latencia p50/p95, ratio de nearduplicates, incidencias por dominio.

Observaciones. Requests ofrece la mejor latencia/throughput en páginas estáticas; Sitemap maximiza cobertura (especialmente PDFs y rutas profundas); Selenium logra completitud de render en portales dinámicos con un coste 3–5× en latencia (se usa sólo cuando es necesario).

Orquestación propuesta. 1) Descubrir con Sitemap si existe; 2) extraer con Requests por defecto (HTML) y pipeline de Documentos para PDFs; 3) reintentar con Selenium sólo en URLs con baja densidad de texto o señales de JS (heurísticas: script/shadowdom, falta de main content); 4) registrar la estrategia en metadatos (web\_strategy) para auditoría y análisis comparativo.

Tabla comparativa (síntesis)

4.2.3 Parámetros clave (y por qué)

chunk\_size = 900 caracteres y overlap = 12%: óptimo experimental → +51% de cobertura semántica con +9.1% de tiempo.

Normalización avanzada: remove\_headers\_footers, normalize\_whitespace, fix\_ocr\_artifacts, preserve\_legal\_structure.

Deduplicación: near\_enabled = True, jaccard\_k = 7, threshold = 0.90.

Seguridad: encryption\_at\_rest = True, control de acceso por documento y audit trail.

4.2.4 Métricas y resultados (ejecuciones reproducibles)

Documentos — Run ID: DOC096 (20250908 18:05 CET)

Documentos procesados: 96

Chunks generados: 5,230

Duración total: 67.4 s

Rendimiento: 77.57 chunks/s

Tasa de error: 0%

Idempotencia (hit rate): 100%

Cómo se obtuvieron y de qué script/artefacto salen

n\_chunks y duration\_s → del artefacto summary.json generado por el pipeline de ingesta de documentos (artefactos en data/processed/runs/documents/DOC096/).

Rendimiento (chunks/s) → cálculo directo con fórmula n\_chunks / duration\_s (a partir de summary.json).

Tasa de error → conteo de entradas de error en summary.json y verificación cruzada en stdout.txt (no se registraron errores).

Idempotency hit rate → skipped\_documents / total\_documents a partir de summary.json.

Coherencia de relaciones BD → validada con tests/verify\_ingestion\_sqlite.py (consistencia Document↔Chunk y metadata completeness).

Reproducibilidad (ejemplo de comando para recomputar rendimiento)

python - <<'PY'

import json, pathlib

p = pathlib.Path('data/processed/runs/documents/DOC-096/summary.json')

s = json.loads(p.read\_text(encoding='utf-8'))

n = s.get('totals',{}).get('chunks') or s.get('summary\_totals',{}).get('chunks')

t = s.get('totals',{}).get('duration\_s') or s.get('summary\_totals',{}).get('duration\_s')

print(round(n/t, 2), 'chunks/s')

PY

Para runs web ver §4.2.6, donde se documentan los resultados por run (incluido Run ID 4001 – 20250910 19:14:39 CET: seed https://www.python.org/, estrategia sitemap) y las métricas asociadas (coverage, chunks/page, nearduplicates, latencia p50/p95).

4.2.5 Corrección OCR específica

Correcciones regex validadas para errores comunes (Art3culo → Artículo, Pr0cedimiento → Procedimiento, etc.), con +40% de mejora en calidad sobre PDFs escaneados. La corrección se aplica antes del chunking para evitar propagar ruido a los embeddings.

4.2.6 Tablas de evaluación a nivel de chunk

(A) Sensibilidad del chunking en documentos (configuraciones evaluadas)

Lectura recomendada: C mejora la cobertura semántica (más evidencia por consulta) con un overhead temporal moderado.

(B) Impacto del overlap en web (nivel chunk)

Conclusión: 10% ofrece el mejor balance entre contexto añadido (+11% chunks) y duplicidad baja.

(C) Resultados por run web (a nivel de chunk)

Nota: Para HTML puro aumenta la densidad de chunks; en mixto HTML+PDF la densidad por página desciende, pero mejora la calidad textual de los PDF al reutilizar el pipeline de documentos.

4.3 Consolidación de ingesta pública/privada

Este contenido se ha integrado en §4.2 (ingesta unificada de documentos y web), donde se describen los pasos, parámetros y métricas con distinción por fuente (A) Documentos, B) Web) y clasificación de seguridad para el routing a vector stores público o privado. Se mantiene esta sección para conservar la numeración del capítulo.

4.4 Trazabilidad, metadatos y cumplimiento

4.4.1 Esquema de metadatos por chunk (y persistencia)

Para qué sirve. El esquema de metadatos estandariza la trazabilidad y la gobernanza del dato a nivel de chunk: permite auditar cada respuesta RAG hasta su origen (documento/URL, página/sección, run\_id), activar filtros de recuperación (por data\_classification, entidad/unidad, tipo y fecha), y mantener la consistencia entre la BD relacional y los vector stores (vía chunk\_id ↔ vector\_id). Además, facilita la reproducibilidad de resultados (comparación entre runs) y el análisis comparativo de estrategias web mediante web\_strategy.

Resultados obtenidos (runs DOC096 y WEBE/WEBB/WEBD).

Auditoría 1:1 entre Chunk.id y vector\_id (FAISS) y metadatos en Chroma; sin inconsistencias referenciales según tests/verify\_ingestion\_sqlite.py.

Cobertura completa de procedencia para los 5.230 chunks del run DOC096 y para los runs web (densidades y neardup documentados en §4.2.6), habilitando filtros por público/privado y por entidad.

Comparabilidad entre estrategias web: web\_strategy (requests/sitemap/selenium) queda registrado por URL/chunk, lo que permite medir impacto en coverage, chunks/página y nearduplicates.

Depuración operativa: checksum y created\_at facilitan idempotencia y detección de cambios reales; las rutas/URLs canónicas evitan duplicidades entre documento y web.

\* Nota FAISS: FAISS no almacena metadatos; se vinculan vía vector\_id = Chunk.id y la metadata reside en la BD relacional (y, opcionalmente, en un sidecar meta.jsonl). En Chroma, la metadata se persiste junto al vector.

Ubicación/flujo:

BD relacional (p.ej., SQLite/PostgreSQL): tablas IngestionRun, Document, Chunk. Claves: Document.id, Chunk.id, Chunk.document\_id, Chunk.index.

Vector stores:

FAISS: almacena sólo embeddings e ids (vector\_id = Chunk.id).

Chroma: almacena embeddings + metadata (campos anteriores bajo metadata.\*).

Cada chunk incluye metadatos mínimos: tipo\_fuente, origen (ruta/URL/endpoint), fecha\_ingesta, entidad, página/sección, clasificación\_seguridad. Esta trazabilidad habilita:

Auditoría de respuestas (RAG) por fuente y fecha.

Filtros de recuperación (por tipo documental, año, unidad administrativa, etc.).

Cumplimiento ENS/CCN: separación de datos, no entrenamiento con inputs de usuario, logging de accesos y justificación de procedencia en UI.

4.5 Optimizaciones del sistema

4.5.1 Caché de embeddings (LRU)

Dónde está implementada. app/rag/embeddings/cache.py (LRU y utilidades) y app/rag/embeddings/provider.py (proveedor de embeddings con wrappers de caché). La indexación la invoca scripts/index\_chunks.py (CLI) y los pipelines de ingesta llaman al proveedor de embeddings mediante services del módulo app/rag/.

Qué hace. Evita recalcular embeddings de textos ya procesados cuando no ha cambiado ni el contenido del chunk ni la configuración del modelo. Esto reduce latencia en reindexaciones incrementales y ahorra CPU/GPU.

Política de invalidación.

Por contenido: si cambia Chunk.checksum (sha256 del texto normalizado), el cache key cambia ⇒ miss forzada.

Por modelo: el cache key incorpora model\_fingerprint = model\_name@version:dim:normalize:pooling. Cambio de modelo o parámetros ⇒ miss global y recálculo.

Por configuración de chunking: el cache key añade embed\_config\_hash (e.g., chunk\_size, overlap, normalizaciones). Cambios ⇒ miss selectiva.

Expulsión LRU: tamaño configurable (p. ej., settings.toml → embeddings.cache.max\_items); los menos usados salen primero.

Métricas instrumentadas. hits, misses, hit\_ratio = hits / (hits + misses), evictions. Se registran por ejecución en cache\_metrics.json (directorio del run) y en stdout.txt.

Resultado real (reindexación incremental, mismos 96 documentos).

Cómo reproducir el hit ratio: volver a ejecutar scripts/index\_chunks.py con el mismo model\_name y sin cambios de texto; comprobar data/processed/runs/documents/DOC096R1/cache\_metrics.json.

Qué hace: evita recalcular embeddings de textos ya procesados (misma firma).

Cómo está implementado: LRUCache + metrics hook (hits, misses, ratio) + invalidación al detectar cambios.

Impacto esperado: reducción de latencia en reindexaciones incrementales y ahorro de CPU/GPU.

4.5.2 Reglas de routing por clasificación

Rutas de indexación hacia private\_faiss/private\_chroma o public\_faiss/public\_chroma según data\_classification y tipo de contenido.

Objetivo: aislamiento de dominios y capacidad de apagar/encender colecciones por motivos de seguridad u operación.

# Capítulo 5. Diccionario de entidades (Gazetteer)

4.6 Base de conocimiento de entidades (fase II)

4.6.1 Diseño y modelo de datos

Entity: name, entity\_type (procedure, regulation, department, place…), canonical\_form, aliases, description, extraction\_method, confidence, frequency.

EntityOccurrence: enlaza entidad↔chunk con ventana de contexto y posiciones.

EntityDictionary: términos curados (definiciones y fuente autoritativa).

4.6.2 Estrategia de extracción

NER (spaCy es\_core\_news\_lg + modelos legales especializados) + patrones (procedimientos: solicitud de, licencia de, autorización para…).

Normalización de variantes y aliases, índices por tipo, frecuencia y co-ocurrencias.

4.6.3 Integración con recuperación

Enriquecimiento de metadatos de chunk con entidades detectadas.

Búsqueda híbrida (texto + entidades) y filtros por tipo de entidad (p. ej., procedures: licencia de obra mayor).

# Capítulo 6. Grafo de conocimiento municipal

4.7 Grafos de conocimiento (fase III)

4.7.1 Alcance

Nodos: Document, Entity, Procedure, Department, Regulation, Service…

Relaciones: REFERENCES, REPLACES, REQUIRES, APPROVES, REGULATES, PROVIDES, DEPENDS\_ON.

4.7.2 Casos de uso

"¿Qué ordenanzas modifican la Ordenanza X?" (REPLACES/REFERENCES)

"Pasos para licencia de obra mayor" (workflows secuenciales)

"¿Quién aprueba el procedimiento Y?" (APPROVES)

"Impacto de derogación de Z" (propagación sobre REGULATES/REPLACES)

# Capítulo 7. Almacenamiento vectorial y comparativa FAISS vs ChromaDB

Capítulo 5. Almacenamiento Vectorial y Comparativa FAISS vs ChromaDB -

5.1. Marco Teórico y Decisiones Técnicas

5.1.1. Vector Stores en Sistemas RAG

Fundamentos Teóricos

Un vector store opera mediante la indexación y recuperación de vectores de embeddings para realizar búsqueda semántica [1]. En sistemas RAG, el almacenamiento vectorial permite seleccionar fragmentos relevantes de un corpus a partir de la similitud entre la representación de la consulta y la de los documentos. Mientras los sistemas tradicionales buscan coincidencias exactas de términos, los vector stores interpretan la intención semántica mediante representaciones matemáticas que capturan el significado del texto en espacios vectoriales multidimensionales.

Cuando un técnico municipal pregunta sobre "licencias para construcción de piscinas", el sistema puede relacionar esta consulta con documentos que hablen de "autorizaciones para obras acuáticas" o "permisos de instalaciones deportivas privadas", demostrando la capacidad de comprensión semántica que diferencia estos sistemas de las búsquedas lexicales tradicionales.

Arquitectura Común Implementada

El sistema desarrollado implementa una arquitectura modular que abstrae las diferencias técnicas entre vector stores mediante un contrato unificado que garantiza intercambiabilidad y trazabilidad completa.

Pipeline de Indexación Unificado

El flujo de indexación común (scripts/index\_chunks.py) consta de las siguientes etapas:

Selección de chunks desde SQLite con filtros configurables (--run-id, --source-id, --limit)

Plan de reindexado con index\_manifest.json mediante cálculo de hash SHA256 del contenido

Embeddings por lotes con batch configurable para optimizar rendimiento

Persistencia en el store elegido (--store faiss|chroma) manteniendo el mismo contrato

Smoke query para validación inmediata del índice construido

Artefactos de Trazabilidad

// index\_meta.json (contrato unificado)

{

"collection": "<str>",

"store": "faiss|chroma",

"model": "sentence-transformers/all-MiniLM-L6-v2",

"dim": 384,

"n\_chunks": "<int>",

"built\_at": "<ISO8601>",

"duration\_sec": "<float>",

"run\_ids": ["<int?>"],

"source\_ids": ["<int?>"],

"checksum": "sha256:<...>",

"notes": "batched=..., ..."

}

5.1.2. Modelos de Embeddings Evaluados: Comparativa Tri-dimensional

La selección del modelo de embeddings constituye una decisión arquitectónica crítica que determina la calidad de la comprensión semántica del sistema RAG. Este trabajo evalúa tres modelos de embeddings implementados mediante Sentence-Transformers, cada uno con características específicas para diferentes contextos de uso en administraciones locales.

Modelos Evaluados

all-MiniLM-L6-v2 (Baseline - 384 dimensiones)

Seleccionado como modelo baseline por su equilibrio óptimo entre calidad semántica y eficiencia computacional [2]. Sus características principales incluyen:

Dimensionalidad: 384 vectores, reduciendo huella de memoria y latencia

Optimización: Entrenado específicamente para tareas de recuperación semántica

Idioma: Rendimiento sólido en español pese a entrenamiento principalmente en inglés

Eficiencia: Procesamiento rápido ideal para despliegues con recursos limitados

intfloat/multilingual-e5-base (Avanzado - 768 dimensiones)

Modelo especializado en contextos multilingües con mayor capacidad representacional [3]:

Dimensionalidad: 768 vectores, duplicando la expresividad semántica

Multilingüismo: Entrenamiento específico en múltiples idiomas incluyendo español

Especialización: Optimizado para documentos administrativos y técnicos

Trade-off: Mayor precisión a costa de incremento computacional y de memoria

BAAI/bge-m3 (Experimental - 1024 dimensiones)

Modelo de última generación con capacidades multifuncionales avanzadas [4][5]:

Dimensionalidad: 1024 vectores, máxima expresividad disponible

Cobertura: Soporte nativo para más de 100 idiomas

Versatilidad: Gestión efectiva de textos cortos y largos

Hibridación: Capacidades de señales densas y dispersas para casos especializados

5.2. Implementación Comparativa

5.2.1. Arquitectura FAISS (IndexFlatIP)

FAISS (Facebook AI Similarity Search) implementa una estrategia de búsqueda exacta optimizada para máxima precisión [6]. La implementación utiliza IndexFlatIP con normalización L2, garantizando recall perfecto (100%) mediante equivalencia matemática con similitud coseno.

Decisiones Técnicas

Índice: IndexFlatIP (producto interno) sobre embeddings normalizados L2

Búsqueda: Exacta con coste computacional O(N·D) lineal

Precisión: Recall 100% garantizado para todas las consultas

Persistencia: index.faiss + ids.npy (mapping posición→chunk\_id)

class FaissStore:

def \_\_init\_\_(self, base\_dir: Path):

self.index = faiss.IndexFlatIP(dim)

self.ids = np.empty((0,), dtype="int64")

def add(self, vectors: np.ndarray, chunk\_ids: np.ndarray):

# Vectores ya normalizados L2 desde embedder

self.index.add(vectors.astype("float32"))

self.ids = np.concatenate([self.ids, chunk\_ids], axis=0)

5.2.2. Arquitectura ChromaDB (HNSW)

ChromaDB implementa búsqueda aproximada mediante algoritmos HNSW (Hierarchical Navigable Small World) [7][8], optimizando latencia a costa de precisión aproximada configurable.

Decisiones Técnicas

Algoritmo: HNSW con métrica coseno nativa

Cliente: PersistentClient con una base de datos por colección

Metadatos: Enriquecimiento automático por chunk para filtrado avanzado

Configuración: hnsw:space="cosine" con embedding\_function=None

class ChromaStore:

def \_\_init\_\_(self, base\_dir: Path, collection\_name: str):

self.client = PersistentClient(path=str(base\_dir))

self.collection = self.client.get\_or\_create\_collection(

name=collection\_name,

metadata={"hnsw:space": "cosine"},

embedding\_function=None

)

Metadatos Enriquecidos

Innovación clave del sistema: incorporación de metadatos estructurados por chunk que habilitan funcionalidades avanzadas:

{

"chunk\_id": "<str>",

"document\_id": "<str>",

"source\_id": "<str>",

"run\_id": "<str|null>",

"title": "<str>",

"url": "<str|null>"

}

5.3. Metodología de Evaluación

5.3.1. Dataset Experimental y Configuración

El dataset experimental se compone de 29,834 chunks de documentos municipales reales indexados mediante source-id 1, representando un corpus representativo de administraciones locales españolas. La evaluación se estructura en tres niveles comparativos según modelo de embeddings, vector store y parámetro k.

Corpus de Validación

El sistema utiliza un CSV de validación estructurado (data/validation/queries.csv) con 50 queries administrativas representativas del dominio municipal español. El dataset incluye ground truth multicapa:

expected\_chunk\_id: Identificadores de chunks esperados

expected\_document\_id: Documentos objetivo por query

expected\_document\_title\_contains: Patrones en títulos documentales

expected\_text\_contains: Fragmentos textuales esperados

5.3.2. Métricas: Recall@k, Latencia, Throughput

Métricas de Calidad de Recuperación

Recall@k por Niveles:

Chunk-level: chunk\_hits / total\_queries\_with\_chunk\_gold

Document-level: doc\_hits / total\_queries\_with\_doc\_gold

Title-contains: title\_hits / total\_queries\_with\_title\_pattern

Text-contains: text\_hits / total\_queries\_with\_text\_pattern

Mean Reciprocal Rank (MRR@k):

python

def mrr\_from\_rank(rank: Optional[int]) -> float:

return 0.0 if (rank is None or rank <= 0) else (1.0 / float(rank))

Métricas de Rendimiento

Latencias (milisegundos):

p50: Mediana de latencias (percentil 50)

p95: Percentil 95 para análisis de cola

mean: Latencia promedio aritmética

5.3.3. Protocolo de Benchmarking

Los experimentos se ejecutaron mediante el framework scripts/comparativa\_recuperadores.py con configuración estándar:

Valores k evaluados: 10, 20, 40

Repeticiones: 1 ejecución por configuración

Hardware: CPU estándar, 16GB RAM

Batch size: 256 (optimizado para memoria)

5.4. Resultados Experimentales

5.4.1. Resultados all-MiniLM-L6-v2 (Baseline)

La evaluación del modelo baseline all-MiniLM-L6-v2 confirma su eficiencia computacional como punto de referencia para sistemas RAG en administraciones locales. Los resultados experimentales se presentan en la Tabla 5.1.

Tabla 5.1: Rendimiento all-MiniLM-L6-v2 en Corpus Municipal (29,834 chunks)

Métricas de Construcción:

Tiempo de indexación: FAISS 740s (12.3min) vs ChromaDB 744s (12.4min)

Dimensionalidad: 384 vectores, optimizando memoria y latencia

Rendimiento: Latencias sub-25ms, adecuadas para uso interactivo

5.4.2. Resultados intfloat/multilingual-e5-base

El modelo multilingual-e5-base demuestra mejoras sustanciales en recall a costa de incremento significativo en latencia y tiempo de procesamiento, como se observa en la Tabla 5.2.

Tabla 5.2: Rendimiento multilingual-e5-base (768 dimensiones)

Análisis Dimensional:

Mejora en recall: 510% incremento vs MiniLM-L6-v2 (12.2% vs 2.0%)

Costo computacional: 3.3x incremento en latencia (69ms vs 21ms)

Tiempo de indexación: FAISS 3895s (65min) vs ChromaDB 3986s (66min)

5.4.3. Resultados BAAI/bge-m3

El modelo BGE-M3 presenta el mayor costo computacional del estudio, con resultados mixtos que cuestionan su aplicabilidad en contextos de recursos limitados, según se detalla en la Tabla 5.3.

Tabla 5.3: Rendimiento BAAI/bge-m3 (1024 dimensiones)

Trade-offs Críticos:

Mejora limitada en recall: 16.3% máximo vs 14.3% de E5-base (+14% relativo)

Latencia prohibitiva: 10x incremento vs MiniLM (208ms vs 21ms)

Costo de indexación: FAISS 12611s (210min) vs ChromaDB 15138s (252min)

5.4.4. Análisis Comparativo Consolidado

Tabla 5.4: Matriz de Eficiencia Comparativa

Conclusiones del Análisis Comparativo:

FAISS vs ChromaDB: Rendimiento equivalente en recall, ChromaDB marginalmente más eficiente en latencia para modelos complejos

Trade-off dimensional: Incremento sub-lineal en calidad vs incremento lineal en costo computacional

Punto de equilibrio: E5-base optimiza la relación calidad/rendimiento para administraciones medianas

5.5. Implicaciones para Administraciones Locales

5.5.1. Limitaciones Metodológicas Identificadas

Los recalls observados (2-16%) y la ausencia completa de coincidencias textuales (text\_rate = 0.0%) revelan limitaciones críticas en la metodología de evaluación que requieren atención inmediata:

Desalineación Dataset-Corpus

Problema 1: Ground Truth Sintético El dataset de validación generado automáticamente presenta desajustes semánticos con el corpus real de documentos municipales, evidenciado por:

Queries artificiales: Extracción directa desde títulos documentales sin contextualización

Expectativas irreales: Ground truth que no refleja consultas naturales de técnicos municipales

Vocabulario específico: Desajuste entre terminología oficial y lenguaje consultivo real

Problema 2: Especificidad del Dominio El corpus administrativo presenta características únicas que condicionan la evaluación:

Jerga técnica municipal: Vocabulario especializado no capturado en modelos generalistas

Estructura documental: Fragmentación que no respeta unidades semánticas administrativas

Variabilidad terminológica: Inconsistencias entre documentos de diferentes departamentos

Implicaciones para la Validez

Estos resultados, aunque técnicamente correctos, no reflejan el rendimiento real que experimentarían los usuarios finales del sistema. La evaluación confirma el funcionamiento técnico de la arquitectura pero requiere:

Dataset de validación realista con consultas reales de técnicos municipales

Ground truth validado por expertos del dominio administrativo

Métricas específicas para documentación gubernamental española

5.5.2. Recomendaciones de Implementación

Configuración por Tipo de Administración

Ayuntamientos Pequeños (< 10K habitantes)

Configuración recomendada: MiniLM-L6-v2 + FAISS

Justificación: Óptima relación recursos/rendimiento con latencias <25ms

Corpus típico: 2K-5K documentos, recall absoluto no crítico

Administraciones Medianas (10K-50K habitantes)

Configuración recomendada: E5-base + ChromaDB

Justificación: Equilibrio recall/latencia (14% recall, 69ms latencia) con escalabilidad

Corpus típico: 10K-30K documentos, beneficio claro del multilingüismo

Administraciones Metropolitanas (>50K habitantes)

Configuración recomendada: E5-base + ChromaDB + filtros por metadatos

Justificación: ChromaDB escala sub-linealmente, filtros departamentales críticos

Corpus típico: >50K documentos, latencia asumible (69ms) por calidad superior

Criterios de Decisión Técnica

Factor 1: Restricciones de Recursos

CPU limitada: MiniLM-L6-v2 obligatorio

Memoria <8GB: FAISS preferible por menor overhead

Latencia crítica: Evitar BGE-M3 (>200ms)

Factor 2: Características del Corpus

Documentos multilingües: E5-base esencial

Corpus >30K docs: ChromaDB recomendado

Actualizaciones frecuentes: FAISS simplifica reindexación

5.5.3. Limitaciones y Escalabilidad

Limitaciones Técnicas Identificadas

Dependencia de Calidad del Corpus

Sensibilidad a OCR: Documentos escaneados degradan recall significativamente

Fragmentación subóptima: Chunks que rompen unidades semánticas administrativas

Variabilidad terminológica: Inconsistencias departamentales impactan recuperación

Limitaciones Operativas

Curva de Aprendizaje

Personal técnico: 2-3 meses para dominio funcional del sistema

Administradores: Conocimiento SQLite y embeddings necesario

Usuarios finales: Formación en formulación de consultas efectivas

Mantenimiento del Sistema

Reindexación periódica: Mensual para documentos activos

Monitorización: Métricas de calidad y latencia en producción

Backup: Estrategias de respaldo para índices críticos

Escalabilidad Futura

Extensiones Recomendadas

Fine-tuning de embeddings en corpus administrativo español

Implementación GraphRAG para relaciones entre entidades

Evaluación con usuarios reales para métricas de satisfacción

Integración BAAI/bge-m3 cuando hardware permita latencias >200ms

# Capítulo 8. Recuperación híbrida y capa de agentes (LightRAG + AGNO)

4.8 Orquestación y monitorización unificadas

4.8.1 Orquestador híbrido

Ejecuta pipelines por componente (privado/público) y, si procede, post-procesos de entidades y grafo.

Registra métricas de rendimiento por componente y end-to-end.

4.8.2 Dashboard de métricas

Privado: 96 documentos → 5,230 chunks, 77.57 ch/s, 0% errores, 100% idempotencia.

Público: 260–500 páginas → 24,671 chunks aprox., 94.9 ch/página (HTML), near-dup 0.11–0.16%.

Entidades/Grafo: planificados con umbrales de aceptación (precisión, recall, latencia) definidos.

# Capítulo 9. Sistema de comparación de LLMs

4.9 Integración de modelos de lenguaje duales y comparador automático

4.9.1 Patrón Strategy para proveedores LLM

OpenAIProvider (API): seguimiento de uso, cálculo de coste por tokens, rate limiting y manejo de errores de servicio.

OllamaProvider (local): control de temperatura y longitud (num\_predict), reintentos y métricas de latencia.

4.9.2 Parámetros de inferencia y control

temperature, top\_p, top\_k, max\_tokens, stop\_sequences.

Política: enviar el mismo prompt enriquecido (consulta + fragmentos RAG) a ambos proveedores y registrar todas las métricas por igual.

4.9.3 Métricas registradas

Rendimiento: latencia end-to-end, tokens/s, disponibilidad.

Económicas (API): coste por consulta (input/output tokens, tarifa por modelo), coste mensual estimado.

Calidad: relevancia contextual (¿cita el fragmento?), completitud (estructura de la respuesta), trazabilidad (menciona fuente/fecha).

Nota: Los experimentos de calidad y coste-beneficio se amplían en el Cap. 6. Este capítulo documenta la implementación y el logging de métricas.

# Capítulo 10. Evaluación integral del sistema

4.10 Métricas de evaluación del sistema (resumen)

4.10.1 Ingesta y calidad de datos

Throughput, error rate, idempotency hit rate, chunk coherence, near-duplicate ratio, metadata completeness.

4.10.2 Recuperación (detallado en Cap. 5)

Recall@k (nivel chunk y nivel documento), MRR@k, latencia (p50/p95), throughput de consultas.

4.10.3 Generación (detallado en Cap. 6)

Calidad percibida (revisión experta), factualidad respaldada, longitud/estructura, trazabilidad (citas a fragmentos), latencia y coste.

4.11 Riesgos, limitaciones y mitigaciones

Sensibilidad a OCR: PDFs escaneados degradan embeddings → mitigado con fix\_ocr\_artifacts y validación manual en casos críticos.

Fragmentación subóptima: chunks que rompen unidades semánticas → mitigado con semantic\_paragraph y ventanas de límite.

Variabilidad terminológica: jerga departamental → mitigado con normalización y futura base de entidades/diccionario.

Desalineación dataset↔consultas (para recall): se requiere ground truth con preguntas reales de técnicos.

4.12 Conclusiones operativas y enlace con Cap. 5–6

La implementación resultante:

Proporciona robustez y trazabilidad en la ingesta (privado/público) con parámetros óptimos reproducibles.

Habilita comparativa controlada entre LLMs (API vs local) con logging uniforme de métricas.

Deja sentadas las bases para entidades y grafos, necesarios para consultas complejas y explicabilidad.

En el Cap. 5 se evalúan vector stores y embeddings (FAISS/ChromaDB; MiniLM/E5/BGE) con Recall@k, MRR y latencia. En el Cap. 6 se presentan los resultados del comparador dual (calidad, coste y latencia), junto con recomendaciones de despliegue para diferentes tamaños de administración.

}

# Capítulo 11. Conclusiones y trabajo futuro

Capítulo 6. Sistema de Comparación de LLMs

6.1. Framework de evaluación dual

6.1.1. Arquitectura del comparador

6.1.2. Integración OpenAI vs Ollama

6.2. Implementación técnica

6.2.1. Proveedor OpenAI (API)

6.2.2. Proveedor local (Ollama)

6.2.3. Métricas automatizadas

6.3. Evaluación experimental

6.3.1. Casos de uso administrativos

6.3.2. Configuración de parámetros

6.3.3. Protocolo de comparación

6.4. Resultados y análisis

6.4.1. Métricas de rendimiento

6.4.2. Calidad de respuestas

6.4.3. Análisis costo-beneficio

6.5. Consideraciones para administraciones públicas

6.5.1. Soberanía vs eficiencia

6.5.2. Cumplimiento normativo

Capítulo 7. Conclusiones y trabajo futuro

7.1. Conclusiones principales

7.1.1. Logros alcanzados

7.1.2. Contribuciones técnicas

7.2. Limitaciones del estudio

7.3. Trabajo futuro

7.3.1. Evaluación con BAAI/bge-m3

7.3.2. Integración GraphRAG

7.3.3. Validación con usuarios reales

7.3.4. Extensión a otras administraciones

ELEMENTOS FINALES

# Bibliografía

BIBLIOGRAFÍA

Referencias bibliográficas:

# Anexos

ANEXOS

ANEXO A.

A.1. Tabla campos SIA

Insertar la tabla de los campos del código SIA

A.2. xxxxx

ANEXO B. Ejemplos de tablas y figuras

B.1. EJEMPLO DE TABLA

Tabla 2. Evolución de uso de los CMS

Fuente: W3techs (Web Technology Surveys)

Notas:

Rafa berlanga

Incvestigación propia

Capítulo 1. Introducción

1.1. Contexto y motivación

1.1.1. Digitalización en administraciones públicas

1.1.2. Retos de acceso a información institucional

1.2. Problemática

1.2.1. Dispersión de fuentes de conocimiento

1.2.2. Limitaciones de sistemas tradicionales

1.3. Objetivos

1.3.1. Objetivo principal (corregir coherencia)

1.3.2. Objetivos específicos

Capítulo 2. Marco teórico

2.1. Chatbots en administración pública

2.1.1. Estado del arte en Europa

2.1.2. Experiencias en España

2.2. Modelos de lenguaje: OpenAI vs modelos locales

2.2.1. Modelos comerciales (OpenAI)

2.2.2. Modelos locales (Ollama)

2.2.3. Soberanía tecnológica

2.3. Recuperación aumentada (RAG)

2.3.1. Arquitectura RAG

2.3.2. Ventajas en contextos institucionales

2.4. Integración de fuentes heterogéneas

2.5. Seguridad y cumplimiento normativo (CCN-TEC 014)

2.6. Grafos de conocimiento (trabajo futuro)

Capítulo 3. Metodología

3.1. Diseño general del sistema RAG

3.1.1. Arquitectura modular

3.1.2. Componentes del pipeline

3.1.3. Principios de trazabilidad

3.2. Framework de comparación dual

3.2.1. Estrategia OpenAI vs locales

3.2.2. Métricas de evaluación

3.3. Protocolo experimental

3.3.1. Configuración controlada

3.3.2. Casos de uso representativos

3.4. Tecnologías utilizadas

3.4.1. Stack tecnológico general

3.4.2. Justificación de selección

Capítulo 4. Sistema de Ingesta y Calidad de Datos

4.1. Arquitectura de ingesta multicanal

4.1.1. Tipología de fuentes heterogéneas

4.1.2. Pipeline por tipo de fuente

4.2. Procesamiento especializado

4.2.1. Documentos (PDF, DOCX, TXT)

4.2.2. Fuentes web (scraping, sitemap)

4.2.3. APIs institucionales

4.2.4. Bases de datos

4.3. Evaluación de calidad de datos

4.3.1. Métricas de completitud

4.3.2. Análisis de errores por fuente

4.3.3. Trazabilidad y metadatos

4.4. Problemática y optimizaciones

4.4.1. Desafíos identificados

4.4.2. Soluciones implementadas

4.4.3. Impacto en pipeline RAG

Capítulo 5. Almacenamiento Vectorial y Comparativa FAISS vs ChromaDB

5.1. Marco teórico y decisiones técnicas

5.1.1. Vector stores en sistemas RAG

5.1.2. Embeddings: all-MiniLM-L6-v2 vs intfloat/multilingual-e5-base

5.2. Implementación comparativa

5.2.1. Arquitectura FAISS (IndexFlatIP)

5.2.2. Arquitectura ChromaDB (HNSW)

5.2.3. Migración y optimización del stack

5.3. Metodología de evaluación

5.3.1. Dataset experimental (166 PDFs municipales)

5.3.2. Métricas: Recall@k, latencia, throughput

5.3.3. Protocolo de benchmarking

5.4. Resultados experimentales

5.4.1. Datos reales all-MiniLM-L6-v2

5.4.2. Datos reales intfloat/multilingual-e5-base

5.4.3. Análisis comparativo de rendimiento

5.5. Implicaciones para administraciones locales

5.5.1. Recomendaciones de uso

5.5.2. Escalabilidad y limitaciones

Capítulo 6. Sistema de Comparación de LLMs

6.1. Framework de evaluación dual

6.1.1. Arquitectura del comparador

6.1.2. Integración OpenAI vs Ollama

6.2. Implementación técnica

6.2.1. Proveedor OpenAI (API)

6.2.2. Proveedor local (Ollama)

6.2.3. Métricas automatizadas

6.3. Evaluación experimental

6.3.1. Casos de uso administrativos

6.3.2. Configuración de parámetros

6.3.3. Protocolo de comparación

6.4. Resultados y análisis

6.4.1. Métricas de rendimiento

6.4.2. Calidad de respuestas

6.4.3. Análisis costo-beneficio

6.5. Consideraciones para administraciones públicas

6.5.1. Soberanía vs eficiencia

6.5.2. Cumplimiento normativo

Capítulo 7. Conclusiones y trabajo futuro

7.1. Conclusiones principales

7.1.1. Logros alcanzados

7.1.2. Contribuciones técnicas

7.2. Limitaciones del estudio

7.3. Trabajo futuro

7.3.1. Evaluación con BAAI/bge-m3

7.3.2. Integración GraphRAG

7.3.3. Validación con usuarios reales

7.3.4. Extensión a otras administraciones

ELEMENTOS FINALES

Bibliografía

Anexos

A.1. Configuraciones técnicas

A.2. Código significativo

A.3. Datos experimentales detallados

A.4. Ejemplos de prompts y respuestas