NeuroByte-IA

Integrantes:

- Christian Michael López Lindao.
- Joaquin Daniel Cabrera Galarza
- Henry Ramos

Fecha publicación: 11/08/2025

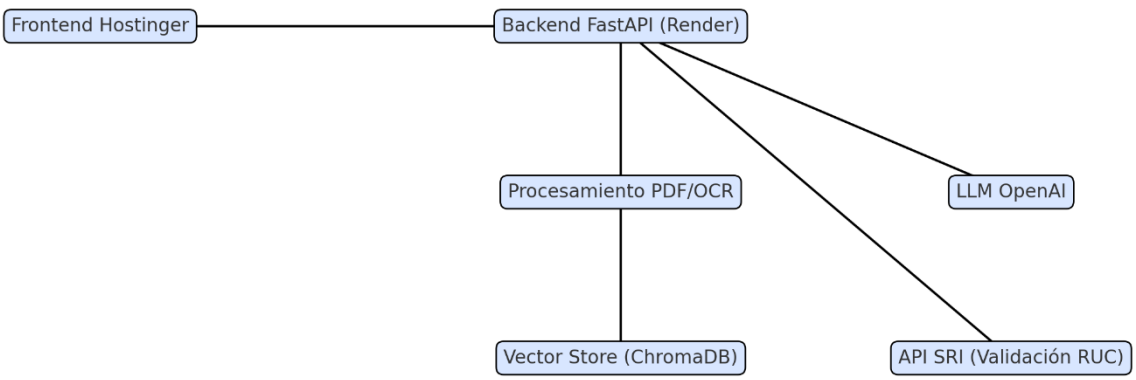
1. Descripción general

Este sistema implementa un modelo **RAG (Retrieval-Augmented Generation)** para el análisis inteligente de documentos de licitación en el proceso de compras publicas.

Su propósito es:

- Indexar documentos PDF (base y subidos por el usuario).
- Recuperar información relevante mediante búsqueda semántica en una base vectorial.
- Generar respuestas naturales y comparativas usando un LLM de OpenAI, con control estricto para evitar alucinaciones.
- Validar datos clave como el RUC frente al servicio oficial del SRI.
- Producir dos salidas principales:
  - respuesta1: narrativa detallada por archivo y comparativo general.
  - respuesta2: JSON de decisión estructurado, persistido por tipo de licitación en /static/tipos/*.json.
- Expone endpoints REST para subir, listar, limpiar, consultar y analizar.

Arquitectura del Sistema RAG para Licitaciones



2. Arquitectura del sistema

2.1 Tecnologías clave

Componente	Tecnología	Uso principal
Framework API	FastAPI, Starlette, uvicorn	Servidor web y endpoints REST
Procesamiento de lenguaje	LangChain, langchain-openai, openai	Embeddings, RAG, consultas al LLM
Vector store	ChromaDB	Almacenamiento y recuperación semántica
Embeddings	OpenAIEmbeddings	Representación vectorial de texto
OCR y PDF	PyMuPDF, Pillow, pytesseract (opcional EasyOCR)	Extracción de texto desde PDFs
HTTP cliente	httpx	Consulta API SRI
Manejo de datos	pydantic, python-multipart	Validación y manejo de formularios multipart

2.2 Flujo de procesamiento

1. **Carga de PDFs base**
    - Indexados en la colección licitaciones_base con metadatos (familia, procedimiento, doc_role) extraídos mediante patrones en kb_config.yaml.
2. **Carga de PDFs de usuario**
    - Guardados en /uploads/pdf/.
    - Procesados para extraer texto (OCR si es necesario).
    - Clasificados (familia, procedimiento) y almacenados en licitaciones_uploads.
3. **Construcción del contexto RAG**
    - Recupera documentos relevantes tanto de la colección base como de uploads.
    - Aplica filtros estrictos o flexibles según configuración (STRICT_BASE).
    - Limita longitud de contexto (CTX_MAX_CHARS_PER_DOC) para optimizar coste y rendimiento.
4. **Extracción de candidatos confiables**
    - Uso de regex y heurísticas para detectar IDs de proceso, plazos, montos, requisitos y cláusulas.

- Separación de secciones base/oferta para minimizar alucinaciones.

5. **Generación de salida con LLM**
   - Se inyecta contexto y candidatos al Prompt Maestro Unificado.
   - El LLM devuelve un JSON validado con:
     - Comparativo
     - Cumplimientos
     - Requisitos
     - Riesgos
     - Recomendaciones
6. **Persistencia**
   - respuesta2 se guarda en /static/tipos/*.json por destino (familia–procedimiento).
   - IDs de proceso se generan únicos usando prefijos calculados.

3. Lógica clave en el código

- **OCR adaptativo**: si EasyOCR falla, usa Tesseract; si OCR está deshabilitado, se omite.
- **Facets y bundles** (KBConfig): clasifican documentos según patrones regex definidos por el usuario.
- **Anti-alucinación**:
  - Contexto limitado a documentos y secciones relevantes.
  - Extracción literal de requisitos.
  - Validación de oferente basada en presencia real en la oferta.
- **Persistencia incremental**: combina resultados nuevos con master.json evitando duplicados.

4. Hallazgos

Fortalezas

- Arquitectura modular con separación clara de funciones.
- Control de contexto para evitar respuestas inventadas.
- Persistencia reutilizable de ChromaDB.
- Integración directa con API pública del SRI.

Áreas de mejora

- Mensajes de error más descriptivos en OCR y lectura PDF.
- Clave de OpenAI debe definirse solo por variable de entorno.

- Endpoint /api/chat podría optimizar el análisis concurrente de múltiples archivos.

5. Pruebas realizadas

Prueba	Descripción	Resultado esperado	Resultado obtenido
Carga base	Indexar PDFs de normativa en licitaciones_base.	Chroma poblada, metadatos correctos.	OK
Carga oferta	Subir PDF de oferta con RUC.	Guardado en /uploads/pdf/, OCR aplicado, clasificación.	OK
Validación RUC	PDF con RUC válido.	Respuesta habilitado=True.	OK
Comparativo	Subir dos ofertas del mismo proceso.	comparativo_texto con diferencias de plazo y monto.	OK
Sin contexto	Pregunta fuera de documentos.	Respuesta vacía o aviso sin datos.	OK

6. Variables de entorno relevantes

- EMBEDDING_MODEL: modelo de embeddings (por defecto text-embedding-3-small).
- CHAT_MODEL: modelo de chat (por defecto gpt-4o-mini).
- CHUNK_SIZE, CHUNK_OVERLAP: configuración de chunking.
- K_BASE, K_UP: cantidad de documentos a recuperar.
- ENABLE_OCR: habilitar/deshabilitar OCR (true/false).
- STRICT_BASE: forzar recuperación estricta por familia/procedimiento.

7. Publicación y despliegue

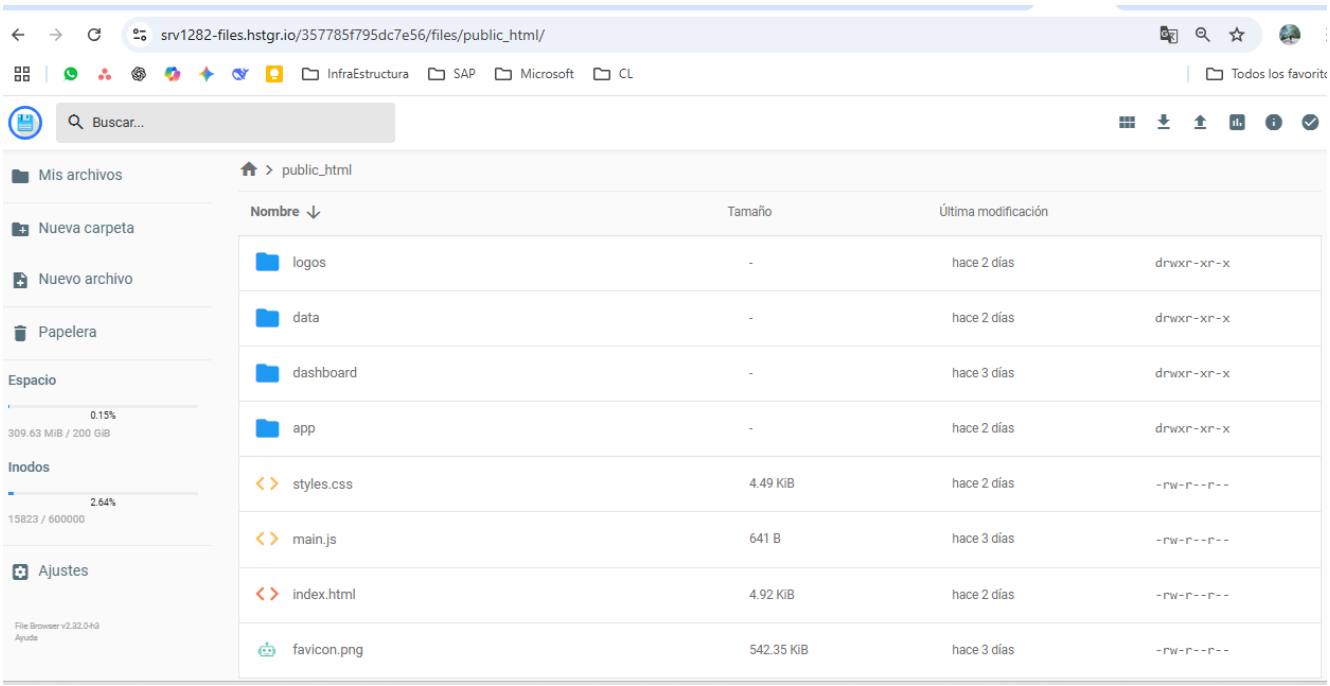
7.1 Entorno de desarrollo

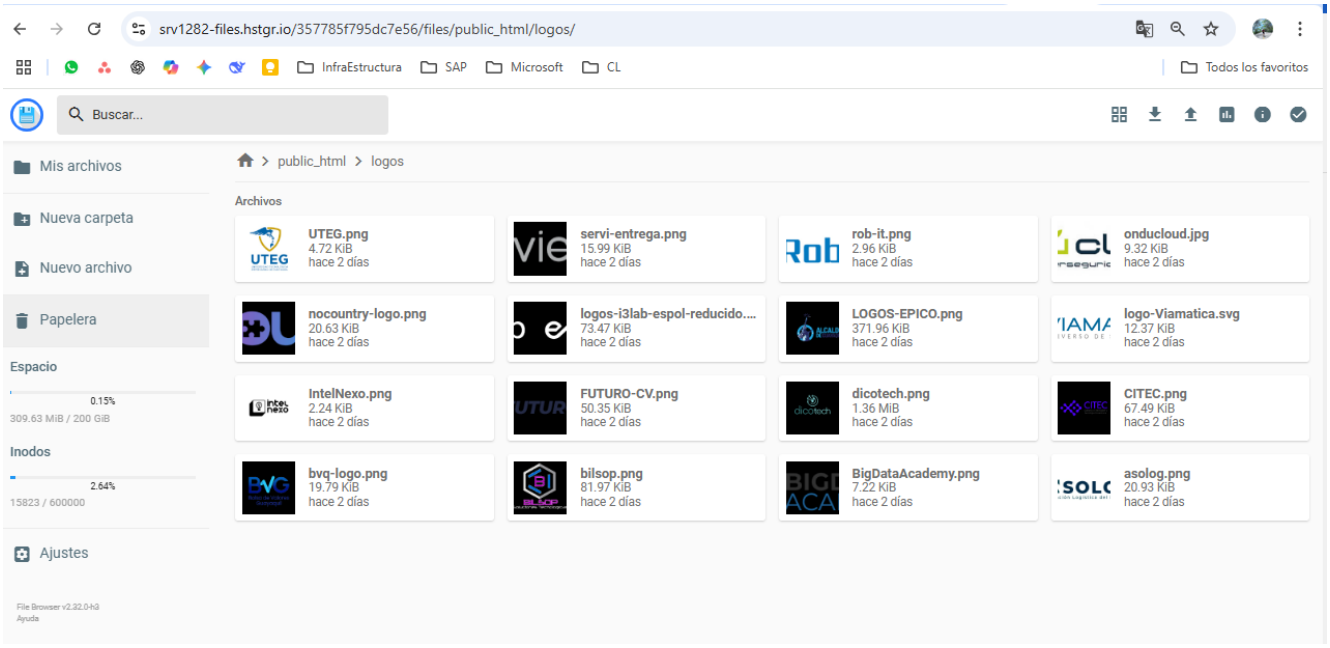
- **Lenguaje**: Python 3.10+
- **Gestión de dependencias**: pip con requirements.txt.

7.2 Publicación en Render

1. Crear repositorio con el código en GitHub.
2. Conectar Render al repositorio.
3. Configurar servicio como Web Service:
    - Runtime: Python.
    - Build Command: pip install -r requirements.txt.
    - Start Command: uvicorn main:app --host 0.0.0.0 --port 10000.
4. Definir variables de entorno (OPENAI_API_KEY, etc.).
5. Desplegar y verificar endpoints.

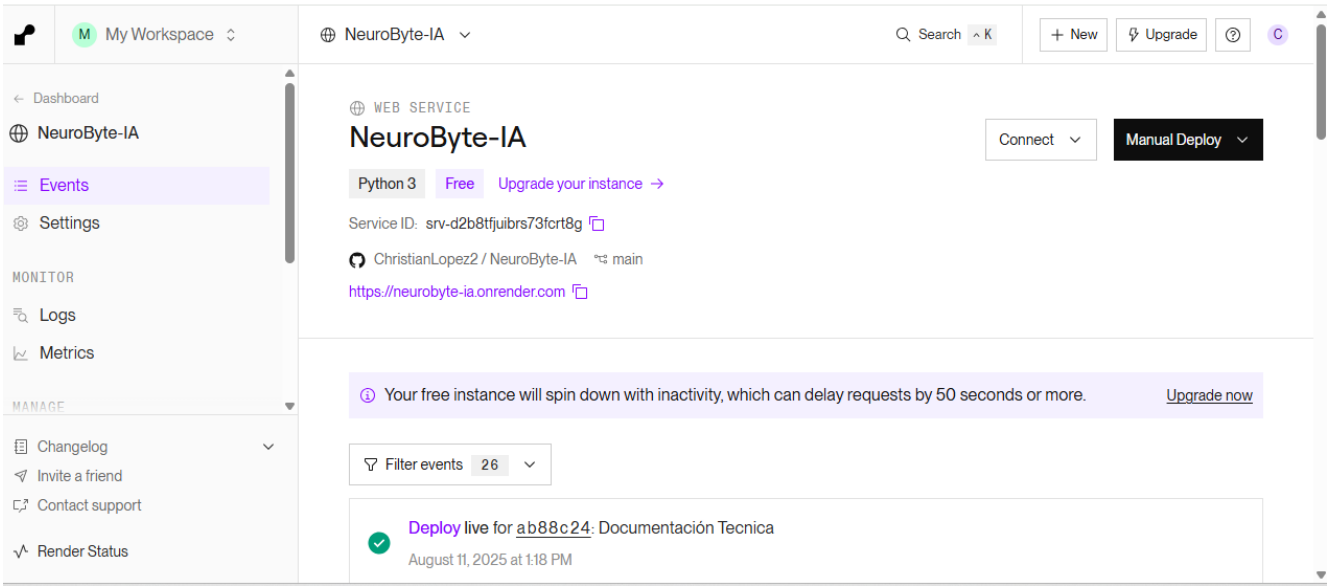
7.3 Publicación en Hostinger

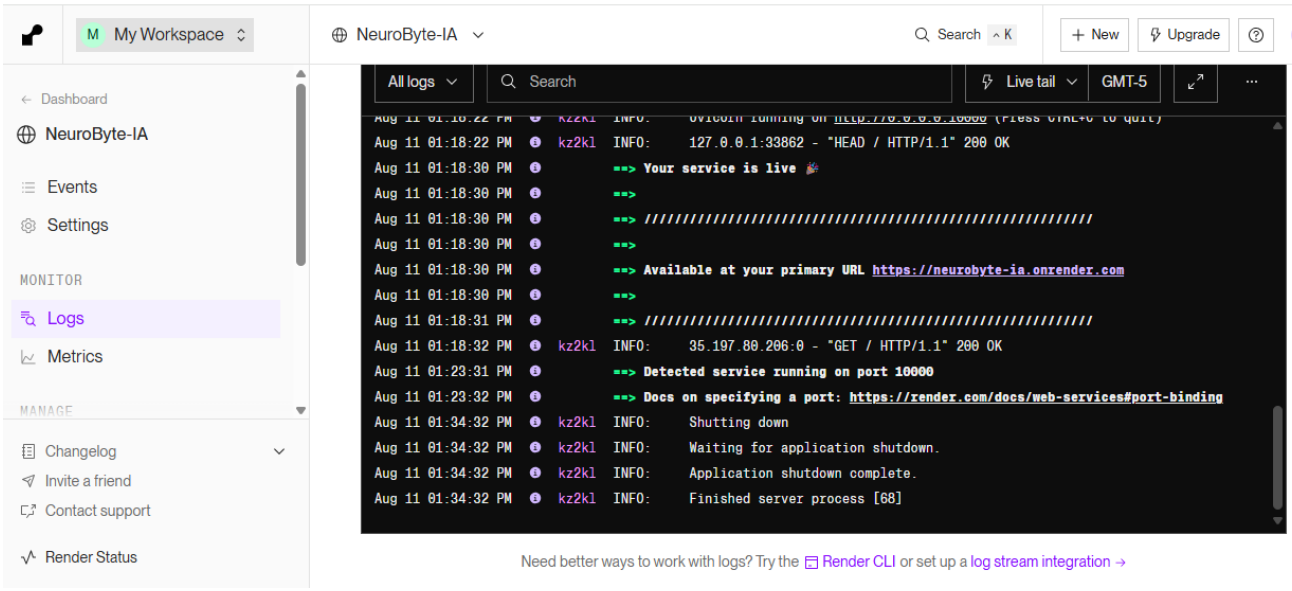
- Usado como frontend hosting para servir los archivos HTML/JS/CSS de la interfaz web.
- Conexión API hacia Render mediante URL pública del backend.
- Configuración HTTPS y optimización de carga estática.
- 

- 

7.4 Integración

- El frontend en Hostinger consume el backend en Render vía endpoints /api/*.
- Control CORS en Fasta configurado con FRONTEND_ORIGIN apuntando al dominio de Hostinger.

- 

- 

8. Conclusión

El sistema RAG para licitaciones integra de manera eficiente recuperación semántica y generación natural de lenguaje, con módulos especializados en:

- OCR adaptativo.
- Clasificación y filtrado de documentos.
- Validación de datos oficiales.
- Persistencia estructurada de resultados.

Neurobyte AI

Ver demo

Optimización Inteligente de Licitaciones

Analiza pliegos, propuestas y contratos con IA. Detecta riesgos legales/técnicos, valida requisitos y compara oferentes en minutos — no semanas.

Probar mi IA | ¿Cómo funciona?

Ahorra días de revisión | Reduce riesgos y sobrecostos | Comparativos claros entre oferentes

Lectura automática de PDFs

NLP para extraer y clasificar secciones clave: legales, técnicas y económicas.

Detección de riesgos

Vacíos, ambigüedades, contradicciones y cláusulas críticas resaltadas.

Comparador de propuestas

Resumen objetivo por cumplimiento, plazos, presupuesto y condiciones.

¿Cómo funciona?

1. **Sube documentos** (pliegos, propuestas, contratos PDF).
2. **La IA clasifica** por secciones y valida puntos clave (RUC incluido).
3. **Detecta alertas** y recomienda mejoras con semáforos y notas.
4. **Compara oferentes** con un dashboard interactivo.

Demo web

Prueba la versión funcional: carga documentos y revisa el análisis en tiempo real.

Ir a la app

- Pliegos y contratos PDF
- Extracción y clasificación automática
- Alertas legales y técnicas
- Dashboard comparativo

Sponsors que hicieron posible esto:

VIAMATICA | servientrega | NoCountry | i3 lab espol

FUTURO | ÉPICO | BIGDATA ACADEMY | BLOCP

SOLOG | Rob-iT | BVG | UTEG

Intelnexo | CITEC | dicotech | ondú cloud

Hecho con ❤ por Neurobyte AI • HackIAthon

↑ Volver arriba