

RAG Portfolio

10 Proyectos de Retrieval-Augmented Generation

De básico a avanzado — PostgreSQL + pgvector + NVIDIA NIM

Python 3.9

FastAPI

PostgreSQL 17

pgvector

NVIDIA NIM

LangChain

SSE Streaming

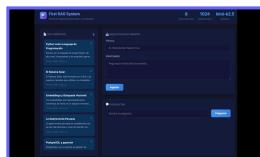
BeautifulSoup4

Manuel Argüelles

Data Engineer / Analytics Engineer

Lima, Perú

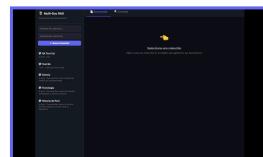
Los 10 Proyectos



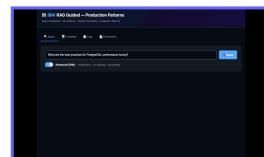
01 Basic



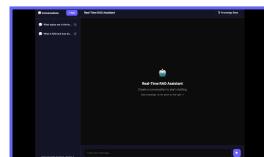
02 PDF



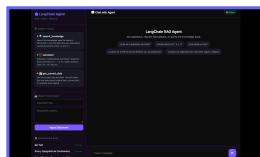
03 Multi-doc



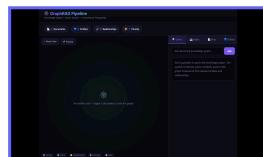
04 IBM



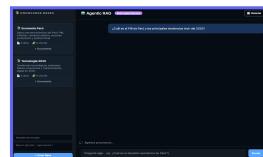
05 Real-time



06 LangChain



07 GraphRAG



08 Agentic



09 Multimodal



10 Research

Febrero 2026

github.com/manuelarguelles/rag-portfolio

Contenido

01

First RAG System

RAG básico desde cero — el patrón fundamental

02

Document Analysis

PDF Processing con chunking y trazabilidad

03

Multi-Document RAG

Colecciones temáticas + búsqueda filtrada

04

IBM RAG Guided

Pipeline enterprise: expansion + reranking + grounding

05

Real-Time Assistant

Streaming SSE token-by-token + memoria

06

LangChain Agent

ReAct Agent con tool selection autónoma

07

GraphRAG

Knowledge Graph + Vector Search híbrido

08

Agentic RAG

4 agentes autónomos colaborando

09

Multimodal RAG

Text + Images en espacio vectorial unificado

10

AI Research Agent

Investigación automatizada end-to-end

¿Qué es RAG?

RAG (Retrieval-Augmented Generation) combina búsqueda de información relevante en una base de conocimiento con generación de respuestas por un LLM. Esto permite responder con información específica y actualizada, reduciendo alucinaciones significativamente.

Pipeline RAG Fundamental:

Pregunta → Embedding (1024d) → Vector Search (pgvector)
→ Top-K Chunks → LLM (Kimi K2.5) → Respuesta Grounded

Stack Tecnológico

PostgreSQL 17 + pgvector

Base de datos vectorial con índice HNSW.
Búsqueda eficiente por similitud coseno.

NVIDIA NIM API (gratis)

nv-embedqa-e5-v5 (1024d) para embeddings.
Kimi K2.5 como LLM de generación.

Python 3.9 + FastAPI

Backend ligero con async. P06 usa Flask.
Sin frameworks pesados.

HTML/CSS/Javascript vanilla

Frontend responsive con dark theme.
Cada proyecto independiente.

Números Clave

10

Proyectos RAG

1024

Dimensiones embedding

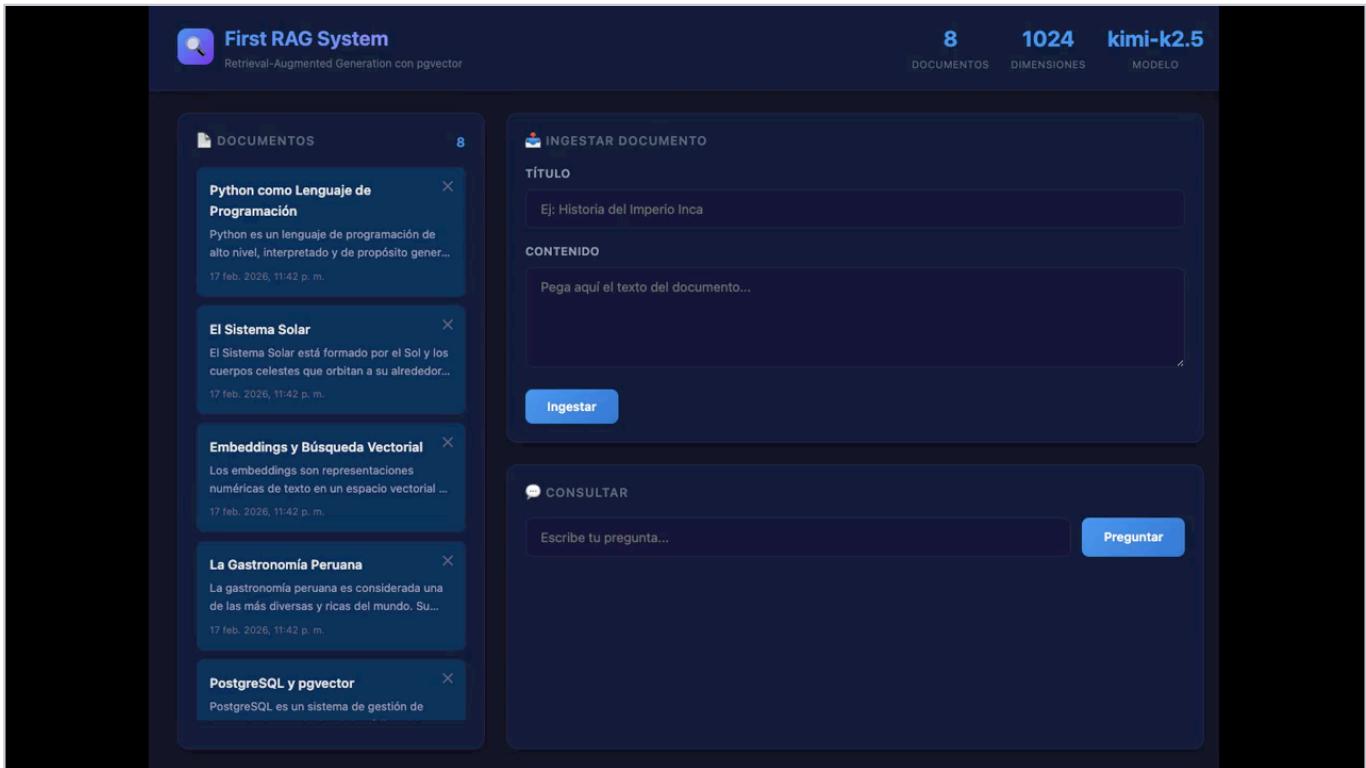
100%

Open source

0

Costo de API

01 — First RAG System



¿Qué es?

RAG básico construido desde cero, sin frameworks. Implementa el patrón fundamental: embedding de la pregunta → búsqueda por similitud coseno en pgvector → contexto al LLM → respuesta grounded con citas. Punto de partida ideal para entender RAG.

Implementación

- ▶ Ingestión de 8 temas (Transformers, pgvector, RAG, SSE, etc.)
- ▶ Búsqueda vectorial con similarity scores visibles
- ▶ Generación con citas al chunk fuente
- ▶ UI tipo chat con panel de resultados

Stack: FastAPI + psycopg2 + pgvector | Puerto: 8000

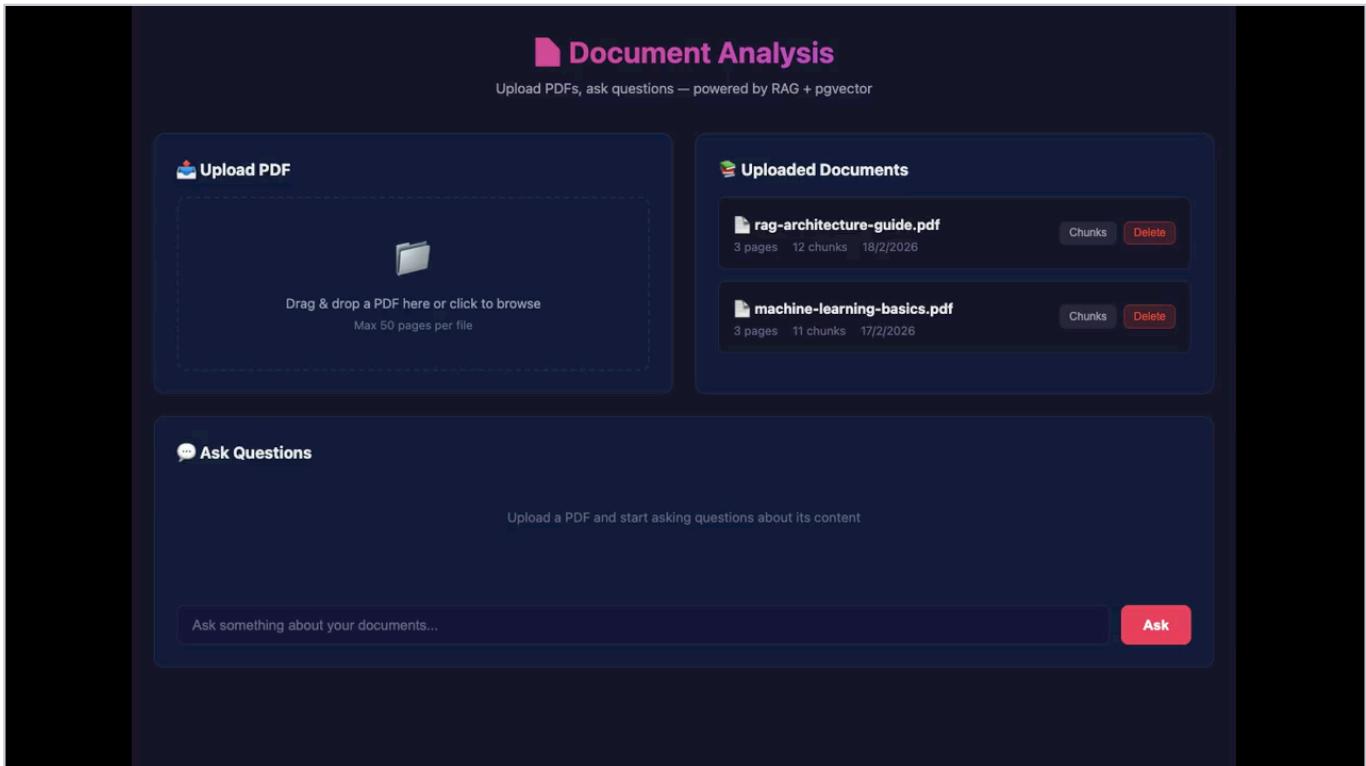
✓ Ventajas

- + Simple y didáctico
- + Sin frameworks
- + Código transparente

✗ Limitaciones

- Sin chunking inteligente
- Sin reranking
- Sin memoria

02 — Document Analysis



¿Qué es?

RAG con procesamiento real de PDFs. Upload → extracción con PyMuPDF → chunking 500 chars / 100 overlap → embedding → búsqueda con citas a la página exacta del PDF.

Implementación

- ▶ Pipeline: drag & drop → extracción → chunking → embedding
- ▶ Trazabilidad hasta página exacta del PDF
- ▶ Chunking con overlap preserva contexto
- ▶ 3 PDFs de ejemplo precargados

Stack: FastAPI + PyMuPDF + pgvector | Puerto: 8002

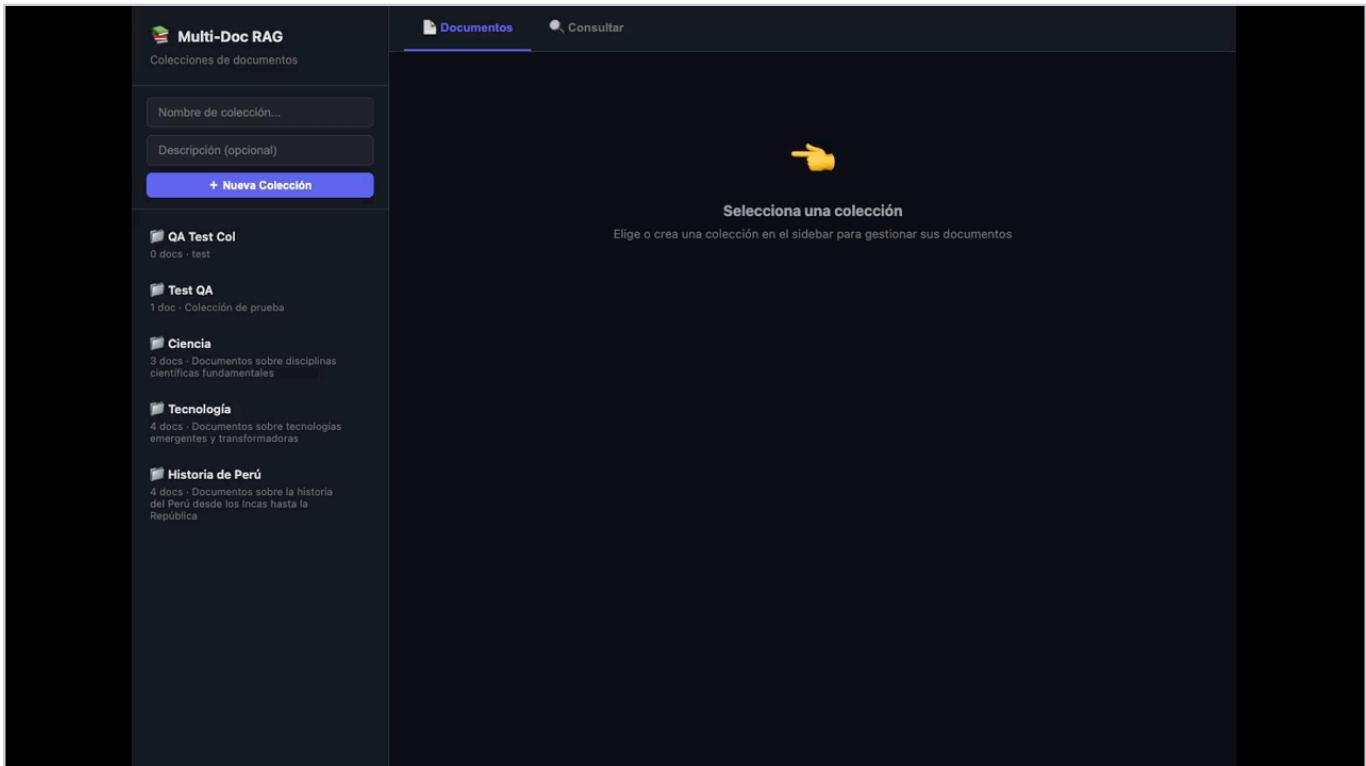
✓ Ventajas

- + Procesamiento real de docs
- + Trazabilidad doc→pág→chunk
- + Overlap preserva contexto

✗ Limitaciones

- Solo PDF
- Sin multi-documento
- Chunking fijo

03 — Multi-Document RAG



¿Qué es?

RAG con múltiples documentos en colecciones temáticas. Búsqueda filtrable por colección o corpus completo. Trazabilidad de 3 niveles: colección → documento → chunk.

Implementación

- ▶ CRUD de colecciones y documentos
- ▶ Búsqueda filtrable por colección
- ▶ 4 colecciones precargadas
- ▶ Trazabilidad de 3 niveles

Stack: FastAPI + psycopg2 + pgvector | Puerto: 8003

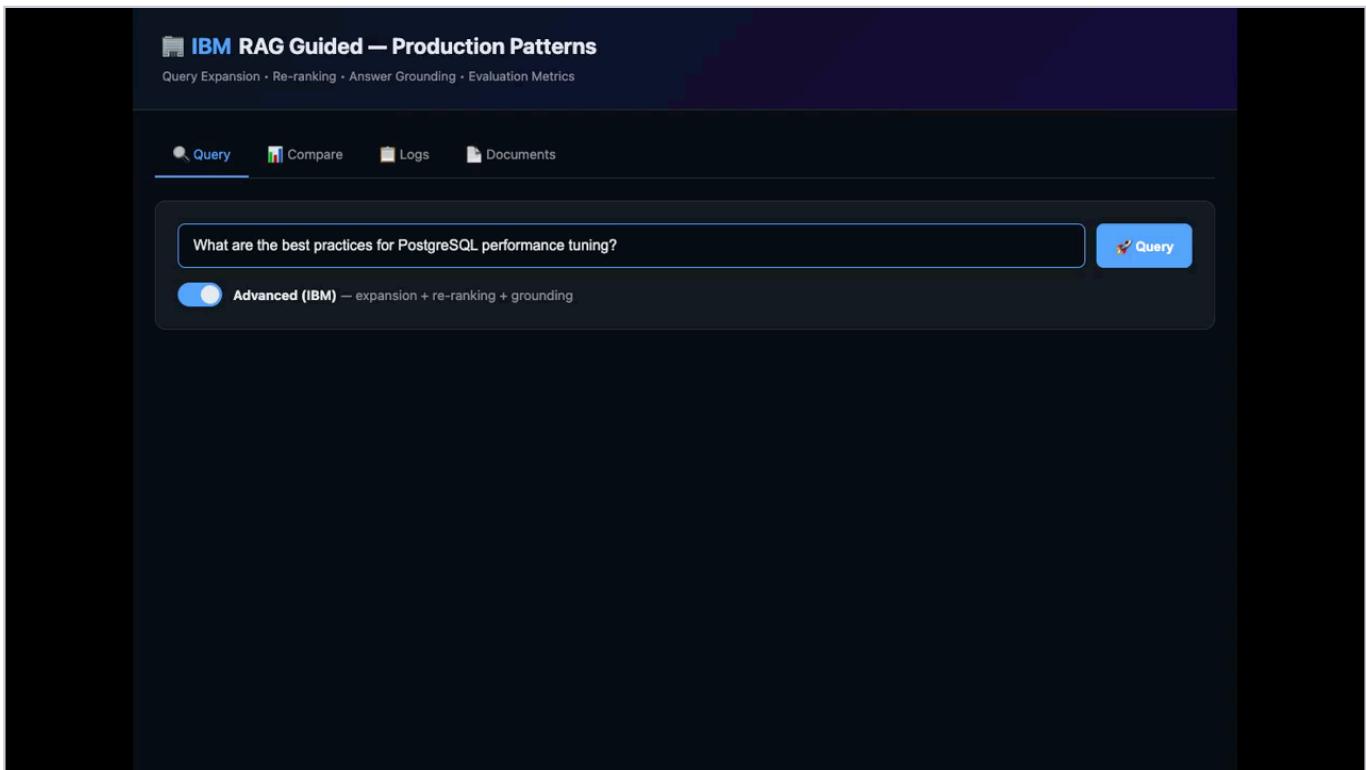
✓ Ventajas

- + Organización por colecciones
- + Búsqueda filtrada
- + Trazabilidad completa

✗ Limitaciones

- Sin reranking
- Sin cross-collection search

04 — IBM RAG Guided



¿Qué es?

Pipeline enterprise: Query Expansion (2-3 variantes), Re-ranking (LLM puntúa 1-10), Grounding Check (verifica fidelidad). Toggle entre modo Simple y Advanced.

Implementación

- ▶ Dos pipelines: Simple vs Advanced
- ▶ Logs visibles de cada etapa del pipeline
- ▶ Métricas de latencia por paso
- ▶ Grounding score de fidelidad

Stack: FastAPI + pgvector (4-5 LLM calls) | Puerto: 8004

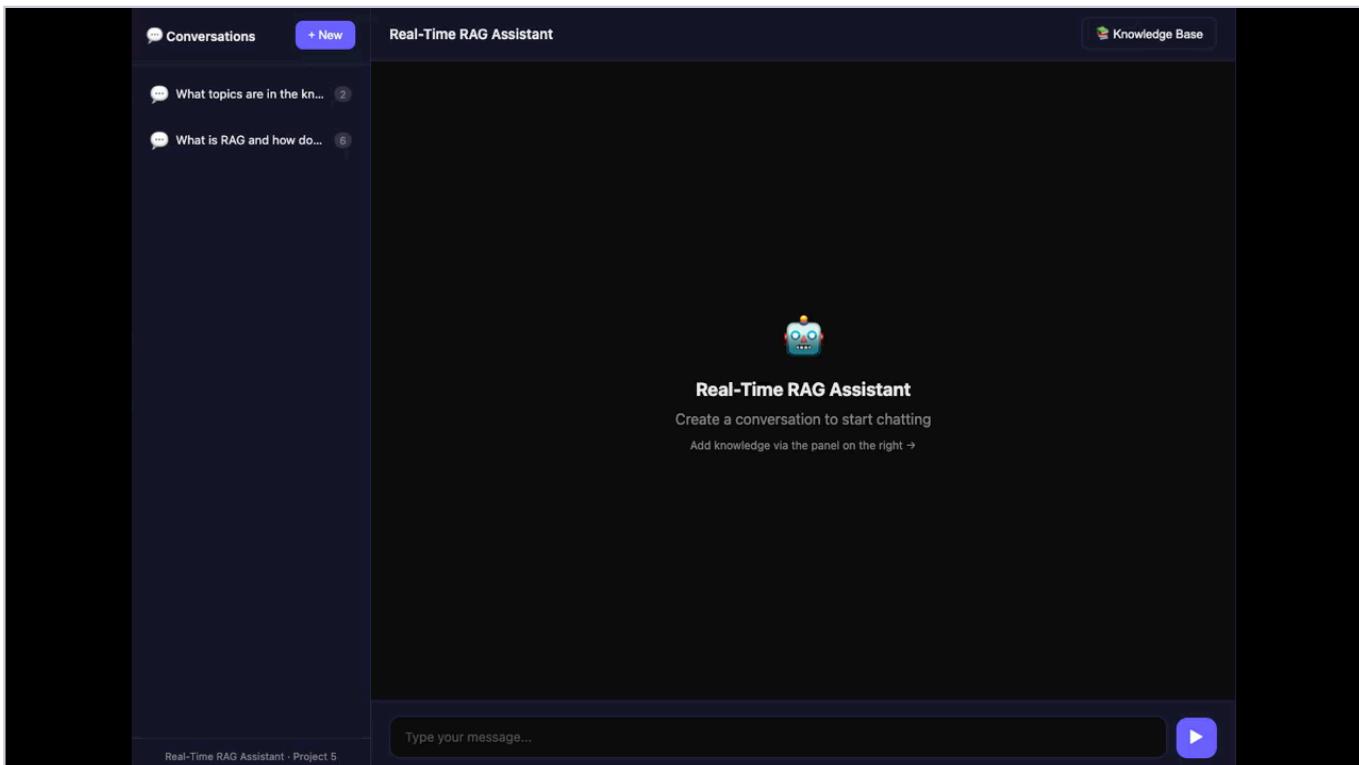
✓ Ventajas

- + Calidad superior
- + Multi-ángulo
- + Reduce alucinaciones

✗ Limitaciones

- Lento (~60-90s)
- Costoso (N LLM calls)
- Debug complejo

05 — Real-Time Assistant



¿Qué es?

Asistente con streaming SSE token-by-token como ChatGPT. Memoria de conversación (5 msgs). Knowledge base editable en caliente sin reiniciar.

Implementación

- ▶ Chat UI estilo ChatGPT con sidebar
- ▶ Streaming vía Server-Sent Events
- ▶ Memoria conversacional (5 msgs)
- ▶ KB editable en caliente

Stack: FastAPI + SSE + pgvector | Puerto: 8005

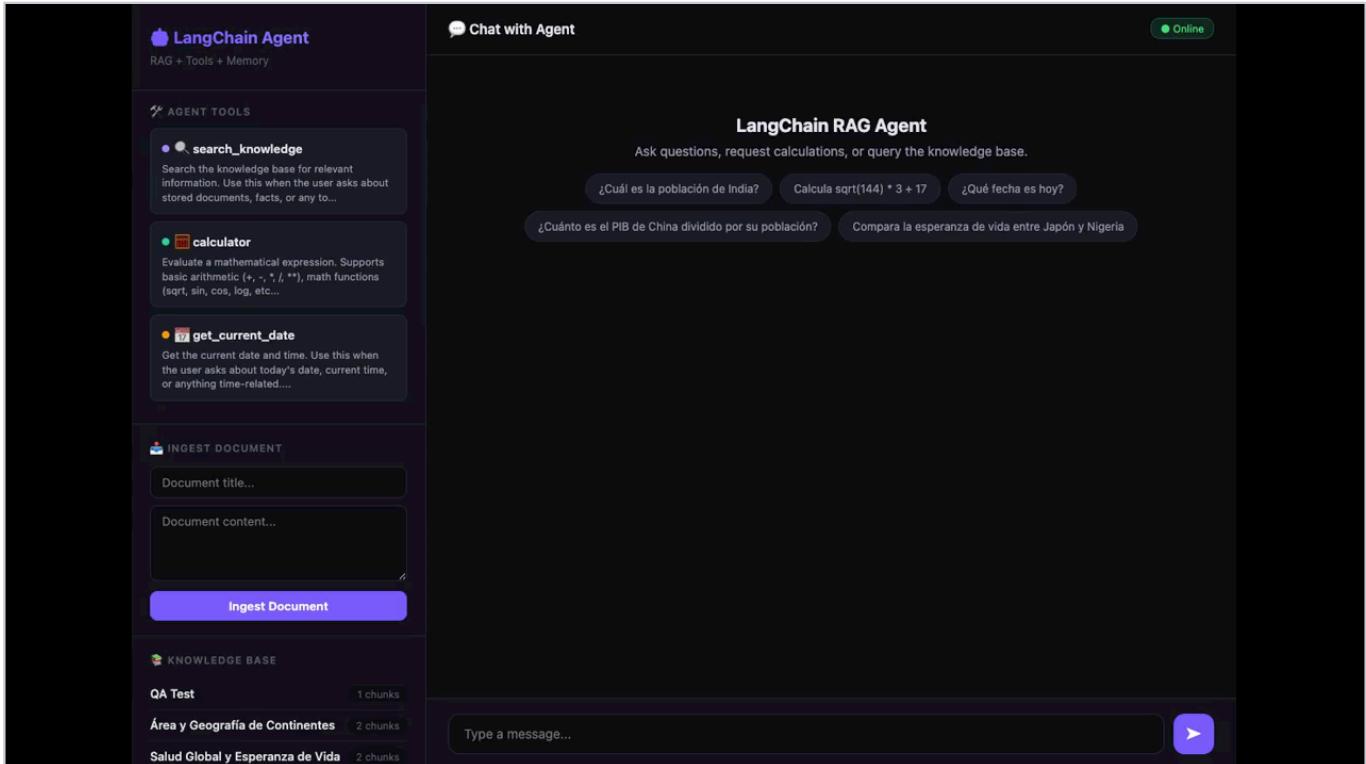
✓ Ventajas

- + UX superior (streaming)
- + Memoria conversacional
- + KB en caliente

✗ Limitaciones

- Solo texto directo
- Memoria limitada

06 — LangChain Agent



¿Qué es?

Agente ReAct con LangChain + LangGraph. Decide autónomamente entre 3 tools: búsqueda vectorial, calculadora, y fecha actual. Custom NvidiaEmbeddings.

Implementación

- ▶ 3 tools: search, calculator, date
- ▶ NvidiaEmbeddings custom (asimétrico)
- ▶ Pipeline de ingestión LangChain
- ▶ 6 documentos precargados

Stack: Flask + LangChain + LangGraph | Puerto: 5006

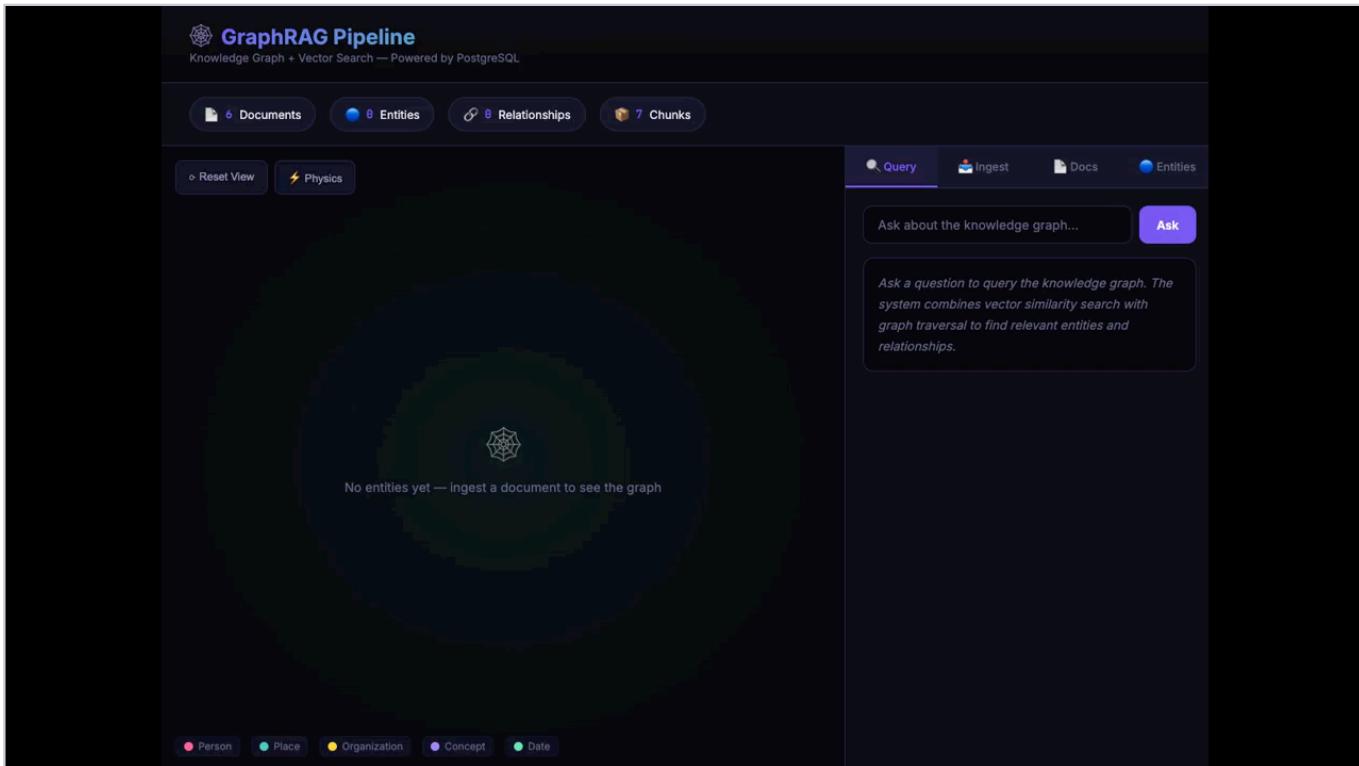
✓ Ventajas

- + Framework maduro
- + Tool selection
- + Patrón ReAct

✗ Limitaciones

- Vendor lock-in
- Overhead
- Debug difícil

07 — GraphRAG



¿Qué es?

Híbrido: vector search + knowledge graph. Extrae entidades y relaciones vía LLM. Multi-hop reasoning implementado en PostgreSQL puro (sin Neo4j).

Implementación

- ▶ Extracción de entidades y relaciones
- ▶ Grafo en PostgreSQL (sin Neo4j/Docker)
- ▶ Búsqueda híbrida: cosine + graph JOINs
- ▶ 6 documentos precargados

Stack: FastAPI + pgvector + SQL JOINs | Puerto: 5007

✓ Ventajas

- + Multi-hop reasoning
- + Todo en PostgreSQL
- + Enriquece contexto

✗ Limitaciones

- Extracción lenta
- Depende del LLM
- Más complejo

08 — Agentic RAG

The screenshot shows the Agentic RAG application interface. On the left, there's a sidebar titled "KNOWLEDGE BASES" with two entries: "Economía Perú" and "Tecnología 2025". Each entry has a description, document count (5 docs), chunk count (5 chunks), and a "+ Documento" button. Below this is a form for creating a new base: "Nombre de la base" and "Descripción (opcional)" with a "+ Crear Base" button. The main area is titled "Agentic RAG Multi-Agent System" and contains a question input field: "¿Cuál es el PIB de Perú y las principales tendencias tech del 2025?". Below the input field, it says "Agentes procesando..." and "Pregunta algo... (ej: ¿Cuál es la situación económica de Perú?)". There's a blue "Enviar" button. A "Historial" button is in the top right corner.

¿Qué es?

4 agentes: Router (planifica), Research (busca), Analyst (analiza), Writer (compone). Python puro — sin LangChain, sin CrewAI. Trace visible de cada decisión.

Implementación

- ▶ 4 agentes especializados
- ▶ 2 knowledge bases temáticas
- ▶ Panel de trace en tiempo real
- ▶ Historial de queries

Stack: FastAPI + pgvector (Python puro) | Puerto: 8008

✓ Ventajas

- + Divide y vencerás
- + Trace visible
- + Sin frameworks

✗ Limitaciones

- 4+ LLM calls (~90s)
- Costoso en tokens
- Orquestación compleja

09 — Multimodal RAG

The screenshot shows the Multimodal RAG application interface. At the top, there are three buttons: 'Multimodal RAG' (selected), 'Text + Images' (disabled), and 'Images Only'. Below these are three summary boxes: '11 Total Items', '6 Text Items', and '5 Images'. A dropdown menu 'Images Only' is open. The main area displays a 3x2 grid of cards:

- Map of Peru**: A simplified map of Peru with a green outline on a dark background. A red dot marks Lima. Below it is the text: 'PERÚ' and 'Map of Peru'. A detailed description follows.
- Python Logo Simplified**: A simplified representation of the Python logo featuring two interlocking shapes in blue and yellow on a dark background. Below it is the text: 'Python' and 'Python Logo Simplified'. A detailed description follows.
- Neural Network Diagram**: A diagram of a simple neural network with four layers: Input, Hidden 1, Hidden 2, and Output. Nodes are purple circles connected by lines. Below it is the text: '5 items' and 'Neural Network Diagram'. A detailed description follows.
- Sunset Palette**: A horizontal gradient color palette showing warm sunset colors from deep purple through magenta, orange, and gold. Below it is the text: 'Sunset Palette' and 'Color Palette: Sunset'. A detailed description follows.
- Entity-Relationship Diagram**: A diagram showing relationships between 'Users', 'Posts', and 'Comments' tables. Arrows indicate foreign key relationships like 'user_id FK'.
- Database Schema Diagram**: A simple entity-relationship diagram showing three database tables: Users, Posts, and Comments. Users have fields for id, name, and email. Posts have id, title...

¿Qué es?

Unifica texto e imágenes en un espacio vectorial. Imágenes indexadas via descripción textual (text-bridge). Galería con filtros y búsqueda semántica unificada.

Implementación

- ▶ 11 items: 6 textos + 5 imágenes
- ▶ 4 vistas: Browse, Upload, Query, Gallery
- ▶ Filtro por tipo (All/Text/Images)
- ▶ Imágenes en BYTEA (PostgreSQL)

Stack: FastAPI + pgvector (text-bridge) | Puerto: 5009

✓ Ventajas

- + Búsqueda unificada
- + UI tipo galería
- + Sin modelo de visión

✗ Limitaciones

- No embedding visual
- Depende de descripción
- Sin image-to-image

10 — AI Research Agent

The screenshot shows a dark-themed user interface for the AI Research Agent. At the top left is the logo and name 'AI Research Agent' with the subtitle 'Investigación automatizada con IA'. Below this is a search bar labeled 'Tema de investigación...' and a large blue button labeled 'Investigar'. To the right of the search bar is a sidebar with a section titled 'Inteligencia Artificial en Latinoamérica...' which includes a progress bar showing 'completed' and '10' items. The main content area starts with a paragraph about the state of AI in Latin America, mentioning its presence in fintech, precision agriculture, and healthcare, and noting a gap between emerging markets and mature ones. This is followed by a section titled 'Hallazgos Principales' with two subsections: 'Adopción y Crecimiento' and 'Innovación Sectorial'. The 'Adopción y Crecimiento' section lists statistics such as 68% of companies using AI, Brazil leading with 40% of startups, and regional investment reaching \$8.5 million. The 'Innovación Sectorial' section highlights agricultural applications like drone monitoring and ML optimization, healthcare improvements, and fintech investments in Nubank AI Labs, Clip AI, and Rappi AI. A final section 'Desafíos' notes the 'Brecha de inversión' (investment gap) where LATAM invests only 12% of what the US does.

de dólares en tecnologías de IA, aunque persiste una brecha significativa con respecto a mercados más maduros. El ecosistema de IA latinoamericano se caracteriza por una fuerte presencia en sectores como fintech, agricultura de precisión y salud, donde la tecnología está generando impactos medibles en productividad y eficiencia. Sin embargo, desafíos como la fuga de talento, la desigualdad en el acceso tecnológico y la necesidad de marcos regulatorios robustos requieren atención urgente. La región muestra un potencial enorme, con casos de éxito que demuestran que la IA puede ser un catalizador de desarrollo sostenible, desde la optimización agrícola hasta la mejora de servicios de salud pública.

Hallazgos Principales

Adopción y Crecimiento

- El 68% de empresas medianas y grandes en LATAM utilizan alguna forma de IA
- Brasil concentra el 40% de startups de IA, seguido por México (22%) y Colombia (12%)
- La inversión regional alcanzó \$8.5 mil millones en 2025, un 62% más que en 2023
- Los sectores fintech (78%), salud (52%) y agritech (47%) lideran la adopción

Innovación Sectorial

- Agricultura:** Drones con IA monitorean 2M+ hectáreas en Brasil; modelos de ML optimizan riego con 30% de ahorro de agua
- Salud:** 120+ hospitales en Brasil usan IA para diagnóstico; México redujo tiempos de espera en urgencias en 40%
- Fintech:** Nubank AI Labs (\$400M), Clip AI (\$180M) y Rappi AI (\$150M) lideran las rondas de inversión

Desafíos

- Brecha de inversión: LATAM invierte solo 12% de lo que invierte EUU

¿Qué es?

Investigación automatizada: genera queries → scrapea web → chunkea → embede → analiza → genera reporte con resumen ejecutivo y hallazgos categorizados.

Implementación

- ▶ Pipeline completo de investigación
- ▶ Demo: 'IA en Latinoamérica 2025'
- ▶ 5 hallazgos categorizados
- ▶ Reporte Markdown estructurado

Stack: FastAPI + httpx + BS4 + pgvector | Puerto: 8010

✓ Ventajas

- + End-to-end automático
- + Hallazgos categorizados
- + Reporte exportable

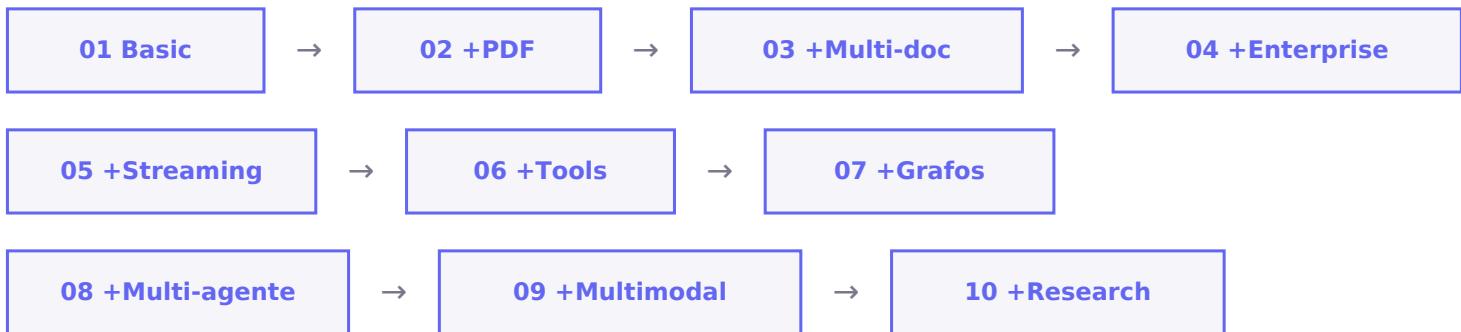
✗ Limitaciones

- Depende de scraping
- Fuentes variables
- Tarda varios minutos

Resumen Comparativo

Cada proyecto agrega complejidad sobre el anterior. El RAG ideal depende del caso de uso — no siempre más complejo es mejor.

Progresión de Complejidad



¿Cuál usar según el caso?

Aprender RAG desde cero	→ 01 First RAG
Documentos internos (PDFs)	→ 02 Document Analysis
Múltiples fuentes organizadas	→ 03 Multi-Document
Producción enterprise	→ 04 IBM RAG Guided
Chatbot conversacional	→ 05 Real-Time Assistant
Múltiples herramientas	→ 06 LangChain Agent
Preguntas sobre relaciones	→ 07 GraphRAG
Consultas complejas	→ 08 Agentic RAG
Contenido mixto (text+images)	→ 09 Multimodal RAG
Investigación autónoma	→ 10 Research Agent

Manuel Argüelles

Data Engineer / Analytics Engineer • Lima, Perú

GitHub: [manuelarguelles](#) • **LinkedIn:** [manuelarguelles](#)