Documentación Técnica - RAG API con FastAPI, Qdrant y Ollama

# 🔰 Introducción

Esta API permite indexar archivos PDF en una base vectorial (Qdrant) utilizando embeddings generados localmente y realizar consultas usando un modelo LLM (como deepseek-r1:1.5b en Ollama), respondiendo preguntas basadas en el contenido de los documentos.

# 🧰 Tecnologías utilizadas

- FastAPI  
- PyPDF  
- SentenceTransformers  
- Qdrant  
- Ollama  
- Docker (opcional)

# 🚀 Endpoints disponibles

## GET /

Devuelve una descripción general de la API y los endpoints disponibles.

Ejemplo de respuesta:

{  
 "mensaje": "RAG API con Qdrant y Ollama",  
 "version": "1.0",  
 "endpoints": {  
 "subir\_pdf": "/subir-pdf/",  
 "consultar": "/consultar/",  
 "health": "/health/"  
 }  
}

## POST /subir-pdf/

Sube y procesa un archivo PDF. Extrae su texto, lo fragmenta, genera embeddings y los almacena en Qdrant.

Parámetros:  
- archivo: Archivo PDF (multipart/form-data)

Ejemplo de respuesta:

{  
 "mensaje": "X fragmentos del archivo 'documento.pdf' insertados correctamente.",  
 "archivo": "documento.pdf",  
 "fragmentos\_procesados": X  
}

## POST /consultar/

Consulta semántica basada en el contenido previamente indexado.

Parámetros (JSON):  
{  
 "pregunta": "¿Cuál es el tema del documento?"  
}

Ejemplo de respuesta:

{  
 "pregunta": "¿Cuál es el tema del documento?",  
 "respuesta": "El documento trata sobre...",  
 "contexto": ["Fragmento 1...", "Fragmento 2..."],  
 "num\_fragmentos\_encontrados": 5  
}

## GET /health/

Verifica el estado de los servicios.

Ejemplo de respuesta:

{  
 "api": "ok",  
 "qdrant": { "status": "ok", "points\_count": 125 },  
 "ollama": { "status": "ok", "models": ["deepseek-r1:1.5b"] },  
 "timestamp": 1717432000  
}

# ⚙️ Flujo general del sistema

1. Usuario sube un PDF.  
2. Se extrae el texto, se fragmenta y se vectoriza.  
3. Los embeddings se almacenan en Qdrant.  
4. Usuario hace una pregunta.  
5. El sistema recupera fragmentos relevantes desde Qdrant.  
6. Esos fragmentos se pasan a Ollama para generar una respuesta.

# 📌 Requisitos

- Python 3.10+  
- Docker  
- Modelos compatibles con Ollama (p. ej. deepseek-r1:1.5b)

# 🐳 Arquitectura sugerida con Docker

- Contenedor app (FastAPI)  
- Contenedor qdrant  
- Contenedor ollama

# 🛠️ Posibles mejoras

- Añadir autenticación.  
- Implementar cache.  
- Dividir la app en módulos.  
- Crear frontend.