Manual Técnico: Agente de IA para RAG en n8n

Juan David Torres Avila

06/06/2025

1. Resumen General

El flujo "RAG 2" implementa un agente de IA con capacidades de Recuperación Aumentada por Generación (RAG), utilizando:

- **n8n** para la orquestación del flujo.
- **Qdrant** como vector store para almacenamiento y recuperación semántica.
- **Ollama** para embeddings y modelos de lenguaje.
- Documentos planos como fuente de conocimiento.

Componentes del Flujo
A continuación, se describen los nodos involucrados, su tipo, función, y conexiones relevantes:

2. Instalación Técnica de Componentes

Este agente RAG requiere la ejecución local de las siguientes herramientas: **Docker**, **Qdrant**, **n8n** y **Ollama**. A continuación se describen los pasos técnicos para su instalación y configuración básica.

2.1. Docker

- Descargar e instalar Docker Desktop desde: `https://www.docker.com/products/docker-desktop`
- Verificar la instalación:

```
docker --version
```

2.2. Qdrant (Base Vectorial)

- Ejecutar Qdrant con Docker:

```
docker run -p 6333:6333 -p 6334:6334 qdrant/qdrant
```

- Esto habilita la API en `localhost:6333` y el dashboard en `localhost:6334`.

2.3. n8n (Orquestador de flujo)

Opción A: Usando Docker

```
docker run -it --rm \
  -p 5678:5678 \
  -v ~/.n8n:/home/node/.n8n \
  n8nio/n8n
```

Opción B: Usando npm

- Requiere Node.js versión 18.x
- Instalar con los siguientes comandos:

```
npm install -g n8n
n8n
```

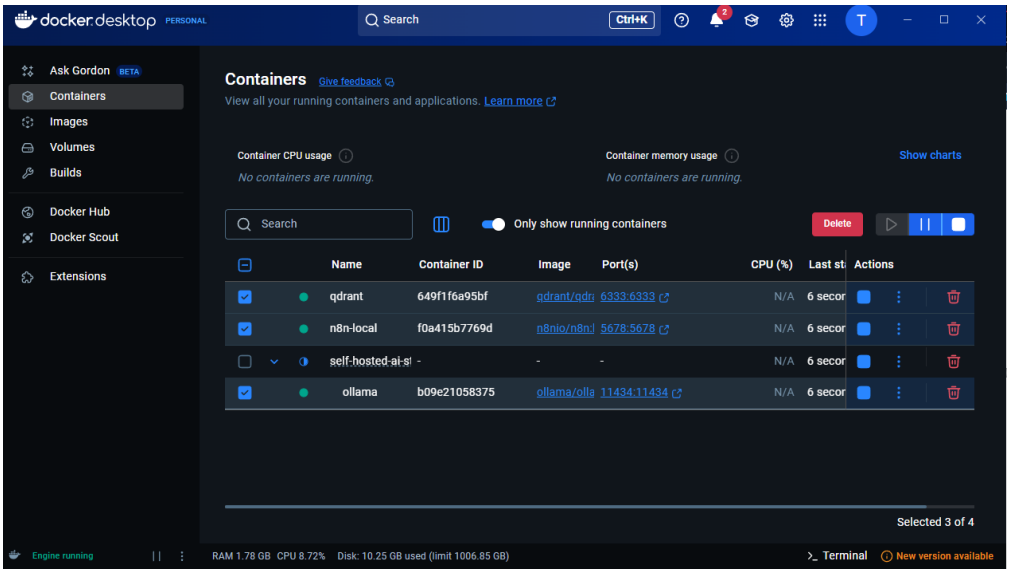
2.4. Ollama (Embeddings y LLM local)

- Descargar desde: `https://ollama.com/download`
- Instalar y ejecutar Ollama.
- Verificar la instalación ejecutando:

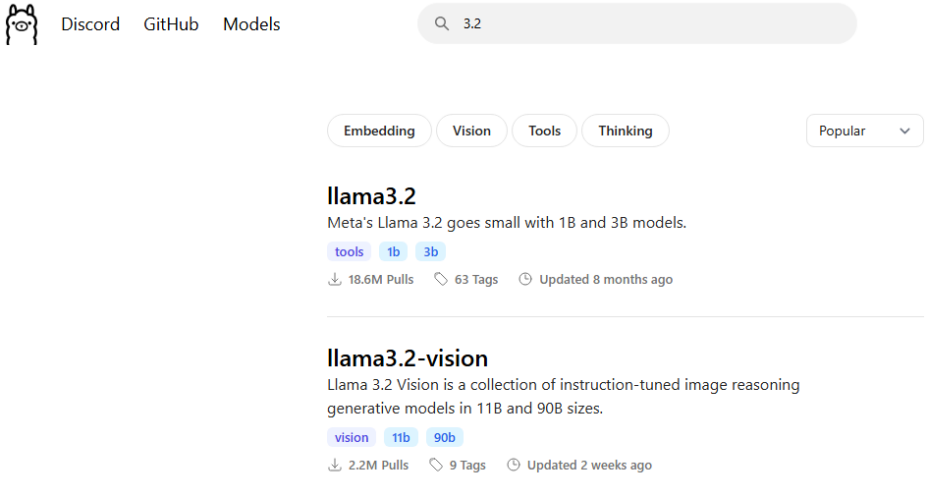
```
ollama run llama3
```

2.5. Validaciones Finales

- Asegurar que docker esta ejecutando los contenedores



- Qdrant debe responder en `http://localhost:6333`
- n8n debe estar accesible en `http://localhost:5678`
- Ollama debe poder ejecutar modelos correctamente desde la terminal, los modelos se pueden descargar desde: `https://ollama.com/search`



2.6. Ingesta y Vectorización de Documentos

- **Schedule Trigger**: Activa el flujo en intervalos programados.
- **Read/Write Files from Disk**:
  - Ruta del archivo: `D:\Descargas\Codigo 2\Prueba-datos\inteligencia_artificial.txt`
- **Embeddings Ollama**: Convierte texto a vectores mediante el modelo `nomic-embed-text`.
- **Recursive Character Text Splitter**: Fragmenta texto en trozos de 800 caracteres con superposición de 100.

- **Default Data Loader**: Carga fragmentos como documentos para su procesamiento posterior.

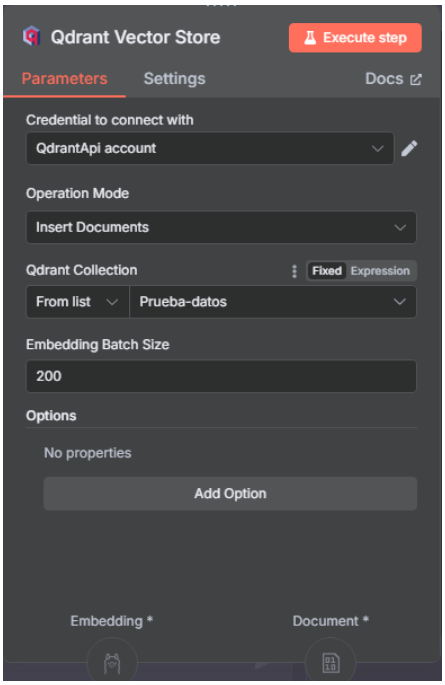
Qdrant Vector Store

Este nodo en el flujo de trabajo de `n8n` permite interactuar directamente con una base de datos vectorial del tipo Qdrant, diseñada para el almacenamiento y recuperación eficiente de vectores de alta dimensión.

- **Colección:** `Prueba-datos`
  La colección especificada, denominada `Prueba-datos`, actúa como un contenedor lógico dentro de Qdrant en el que se almacenan los vectores. Cada colección puede tener configuraciones particulares como el tamaño de los vectores, métodos de indexación y metadatos asociados.
- **Operación:** Inserción de vectores
  La operación configurada en este nodo es la **inserción** de vectores. Esto significa que el nodo está preparado para recibir vectores generados previamente (por ejemplo, a partir de textos embebidos mediante modelos de lenguaje) y almacenarlos en la colección indicada.

  Cada vector que se inserta puede contener:
  - Un identificador único (`id`)
  - El vector numérico propiamente dicho
  - Metadatos opcionales, como por ejemplo el texto original o etiquetas de clasificación

  Esta funcionalidad es clave para la construcción de aplicaciones basadas en búsqueda semántica, recuperación aumentada por conocimiento (RAG), o sistemas de recomendación basados en similitud vectorial.



2.7. Recepción de Preguntas vía Webhook

- **When chat message received**:
  - Activa webhook al recibir un mensaje.
  - Webhook ID: generado automáticamente por n8n.
- **Edit Fields (Set)**:
  - Extrae y asigna variables:

```
chatInput = $json?.chatInput || $json.body.chatInput
sessionId = $json?.sessionId || $json.body.sessionId
```

2.8. Agente IA con RAG

El agente de inteligencia artificial implementado emplea la arquitectura RAG (Retrieval-Augmented Generation), diseñada para combinar la recuperación de información semántica con generación de texto mediante un modelo de lenguaje. La implementación está basada en el entorno n8n, integrando nodos de procesamiento y herramientas externas.

- **AI Agent**: Nodo central que orquesta el flujo de trabajo. Se encarga de gestionar las consultas del usuario, coordinar el uso del modelo de lenguaje, la memoria y las herramientas conectadas. Actúa como punto de integración entre los diferentes módulos.
  - LLM: **Ollama Chat Model**
  - Memoria: **Simple Memory**
  - Herramienta: **Answer questions with a vector store**

- **Ollama Chat Model**: Modelo de lenguaje utilizado para generar respuestas conversacionales. Se conecta al agente como su modelo principal de generación. Utiliza la API de Ollama y ofrece capacidades de interacción natural en lenguaje humano.
- **Simple Memory**: Sistema de memoria de ventana (buffer) que mantiene el historial de conversación para preservar contexto entre turnos. Utiliza una clave de sesión fija (`fafcfe91178e4ccd850c21cb8b5d27d4`) para asegurar coherencia durante la sesión.
- **Qdrant Vector Store1**: Almacén vectorial conectado al agente mediante la herramienta de recuperación. Contiene una colección denominada `ia_txt` que permite la búsqueda semántica de información relevante, previamente embebida y fragmentada para facilitar su indexación.
- **Embeddings Ollama1**: Servicio que convierte la entrada del usuario en vectores de embeddings utilizando el modelo `nomic-embed-text:latest`. Estos vectores son luego utilizados para la búsqueda en el vector store.
- **Ollama Model**: Modelo de lenguaje adicional (diferente del modelo conversacional principal) utilizado dentro de la herramienta de vector store para sintetizar respuestas a partir del contexto recuperado.
- **Answer questions with a vector store**: Herramienta conectada al agente que permite recuperar fragmentos relevantes desde Qdrant basándose en la similitud semántica con la consulta del usuario. Usa el modelo de embeddings y el motor de generación para producir respuestas fundamentadas en información previa.

3. Conexiones Clave del Flujo

- Lectura de archivo → Vectorización → Qdrant
- Mensaje vía Webhook → Extracción de campos → Agente IA
- Agente IA combina:
  - Consulta a Qdrant con embeddings
  - LLM Ollama
  - Memoria de conversación

4. Datos Técnicos Importantes

- **ChunkSize**: 800 caracteres
- **Overlap**: 100 caracteres
- **Modelo de Embeddings**: `nomic-embed-text`
- **LLMs**: Ollama con configuración por defecto
- **Cargas a Qdrant**:
  - `Prueba-datos`: carga inicial del documento
  - `ia_txt`: utilizada en tiempo de consulta

5. Consideraciones Finales

- Asegúrese que Ollama esté ejecutando el modelo requerido localmente.
- Las colecciones deben existir previamente en Qdrant.
- Este flujo puede adaptarse fácilmente a múltiples documentos y colecciones.