Documentación del Proyecto: RAG en n8n local

**Objetivo del Proyecto:**

Este workflow permite la solución de preguntas realizadas por el usuario sobre contenido en archivos tipo PDF y DOCS especificados por el mismo, todo esto usando recuperación semántica con vectores en Supabase y un LLM (determinado a lo largo del documento).

**Tecnologías utilizadas:**

**- n8n:** Automatización de flujos.

**-Docker:** Plataforma de código abierto que permite a los desarrolladores empaquetar y distribuir aplicaciones en contenedores, que son unidades de software estandarizadas que incluyen el código y todas sus dependencias necesarias para su ejecución en cualquier entorno.

**-WSL:** WSL (Windows Subsystem for Linux) es una característica de Windows que permite ejecutar un entorno Linux directamente en tu sistema Windows, sin necesidad de una máquina virtual o arranque dual.

**- Supabase:** Base de datos con pgvector (extensión para Postgres que permite el almacenamiento eficiente y la búsqueda de similitud de datos vectoriales de alta dimensión) para búsqueda semántica.

**- Groq:** Motor de ejecución para modelos LLM.

**- Google Drive:** Fuente de archivos.

**- Ollama:** Herramienta que permite la ejecución de modelos LLM localmente. En este caso se utilizó el modelo de incrustación de texto “nomic-embedding-text” para los embeddings en diferentes puntos del RAG.

**Flujo general**

**Descripción de los nodos involucrados:**

**Nodo: Archivo creado/Archivo actualizado (Google drive)**

Este nodo se encarga de detector cambios en una carpeta en específico, ya sea que se ingrese un nuevo archivo o se actualice el contenido de uno ya creado. Cuenta con un poll time de 3 minutos, lo que significa que mientras el workflow se encuentre activo este nodo revisara cada 3 minutos si se cumple alguna de estas dos condiciones.

**Nodo: Variables (n8n)**

Este nodo se encarga de guardar información extraída del nodo previo, esta será utilizada en nodos posteriores tanto para el reconocimiento de archivos o la inclusión de metadatos en los vectores.

**Nodo: Eliminar coincidencias (Supabase)**

Aprovechando la extracción de información del nodo “variables”, este nodo se encarga de comparar el id del archivo creado con los ids en los metadatos guardados en los vectores dentro de la tabla en supabase, en caso de encontrar una coincidencia borra los registros previos que contengan este número de identificación, este nodo está pensado para no conservar información incorrecta en el caso de haber un archivo actualizado.

**Nodo: Descarga de archivo (Google drive)**

Este nodo utiliza la información de nodos anteriores y descarga el archivo que corresponda indicado por su id.

**Nodo: Switch (n8n)**

Este nodo cumple la función que indica su nombre, es un switch, un cumulo de condicionales, compara el mimetype extraído por medio del nodo “variables” con un estándar general dado por Google, básicamente detecta que tipo de archivo es por medio de su extensión, tomando diferente camino para cada tipo, además cuenta con un fallback que lleva los archivos por un camino específico en caso de no ser ninguno de los dos tipos, una especie de “else”.

**Nota:** NO UTILIZAR CON ARCHIVOS EXCEL.

**Nodo: Extraer texto de PDF (n8n)**

Este nodo extrae datos de un archivo de formato binario y los convierte a JSON, que luego puede ser fácilmente manipulado por el resto del workflow.

**Nodo: Extraer texto de DOCS (n8n)**

Este nodo realiza la misma función que el nodo previamente explicado, la diferencia es el tipo de archivo con el que lo realiza, en este caso documentos de Google.

**Nodo: Almacenar en supabase (Supabase)**

Este nodo se encarga de insertar los vectores conseguidos mediante los nodos adyacentes que serán explicados a continuación:

**Nodo: Embeddings (Ollama)**

Este nodo se encarga de la transformación del lenguaje natural a vectores, en este caso el texto de los archivos, un embedding en pocas palabras.

**Nodo: Default Data Loader(n8n)**

Este nodo se encarga de cargar los archivos de datos binarios o datos JSON para el almacén en vectores.

**Nodo: Recursive carácter text splitter (n8n)**

Este nodo se encarga de dividir los datos del documento recursivamente para mantener todos los párrafos, frases y palabras juntos el mayor tiempo posible.

**Nodo: Mover archivo a procesados (Google Drive)**

Este nodo se encarga de llevar el archivo tratado a una carpeta auxiliar para evitar un error de sobreposición generado en el nodo inicial de Google Drive.

**RAG**

**(AI Agent)**

**Descripción de los nodos involucrados:**

**Nodo: RAG (n8n)**

Este nodo acumula los nodos encargados de llamar al LLM, llamar al nodo que recupera la información de los elementos, etc. Además, puede recibir un “System message”, un prompt que se le enviara al LLM que le debe ayudar a contextualizar su funcionalidad, también puede afectar la regularidad con la que usa sus herramientas.

**Nodo: Groq chat model (Groq)**

Este nodo es el encargado de llamar al LLM **(qwen-qwq-32b,** modelo de razonamiento de inteligencia artificial desarrollado por Alibaba. Cuenta con un límite de 131.072 tokens y está diseñado para realizar tareas de razonamiento complejas, como resolver problemas matemáticos, de programación, y generar código.**)** y generar la respuesta final que recibirá el usuario.

**Nodo: Memory (Postgres)**

Este nodo se encarga de almacenar los prompts y respuestas, actúa como una memoria que recuerda conversaciones pasadas, guarda la conversación como historial.

**Nodo: Responde preguntas con información de vectores (n8n)**

Este nodo es una herramienta que le permite a el agente “resumir” los resultados y responder preguntas basado en chunks de un almacenamiento de vectores.

**Nodo: Almacenamiento de vectores (Supabase)**

Este nodo permite el acceso a la tabla utilizada para recuperar los vectores que se consideren útiles.

**Nodo: Groq chat model1 (Groq)**

Este nodo utiliza un LLM para recompilar los vectores y analizar la información que considere o no relevante para enviársela devuelta al modelo principal

**Nodo: Embeddings (n8n)**

Este nodo convierte el prompt del usuario en un vector para compararlo con los existentes en supabase

**Manual de usuario**

**Requisitos previos:**

* Tener instalado n8n ya sea dentro de Docker o el software de su preferencia, o en su defecto la versión cloud.
* Tener instalado el modelo de incrustación de texto “nomic-embedding-text” de Ollama para los embeddings.
* Configurar las credenciales mencionadas previamente en el documento como tecnologías (Google Drive, Supabase, Groq, Ollama).
* Descargar el archivo en formato JSON que contiene el workflow encontrado en el repositorio de GitHub.
* Crear dos carpetas en Google Drive, la primera donde se irán subiendo los archivos que se quieran relacionar con el RAG y la segunda es una carpeta auxiliar donde se llevaran estos archivos como alternativa para solucionar un error dado en los nodos de este servicio.
* Crear un nuevo workflow dentro de n8n y dirigirse a la esquina superior derecha donde se encuentra la opción de importar archivo.
* Importar el archivo.
* Activar el workflow.
* Subir el archivo que deseas procesar a la carpeta de Google Drive configurada. **Importante:** subir un archivo a la vez, ya que, si se detectan varios simultáneamente, solo se procesará el último.
* Finalmente, en el chat se pueden realizar las consultas sobre el documento y el LLM se encargará de responderlas.

**Notas:**

* Apartados como el modelo que se utiliza en los embeddings o el LLM, o incluso el vector store pueden ser modificados a gusto propio, tomar en cuenta adecuaciones en caso de hacerlo.
* El workflow está diseñado para ser utilizado por archivos independientes separados por un periodo de tiempo entre ellos, en caso de subir varios archivos al mismo tiempo solo se tomará el ultimo integrado.