Agente inteligente de documentación Utb

Primero montamos n8n en local usando Docker

Creamos el Docker-compose.yml:

Texto

El contenido generado por IA puede ser incorrecto.

Levantamos el contenedor ejecutando docker-compose up -d en la terminal

Ahora vamos a <http://localhost:5678> y creamos el flujo para subir documentos a la base de datos vectorial.

**Flujo para subir documentos**

Primero creamos un nodo de código que liste los PDFs.

Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.

Texto

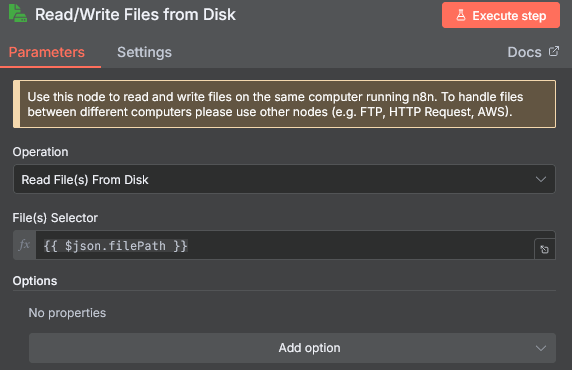
El contenido generado por IA puede ser incorrecto.

Los archivos tienen que estar en la carpeta donde se creó el Docker.

El segundo nodo es un nodo de lectura que va a leer los PDFs que listamos en el nodo de código.

Imagen de la pantalla de un celular con letras

El contenido generado por IA puede ser incorrecto.



Ahora creamos un nodo de extract from file, que convierte el PDF binario en texto estructurado, línea por línea o por páginas. El resultado es texto plano que ya puede usarse en otros nodos (ej. embeddings, búsqueda, etc.).

Imagen de la pantalla de un celular con letras

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ahora usamos un nodo de vector store (en este caso qdrant) conectado a un nodo embedding y a un nodo que carga los documentos que ya estaban en el flujo. El nodo de embedding genera embeddings vectoriales del texto y se guardan el base de datos vectoriales, (tenemos que conectar las credenciales de qdrant a el nodo).

Interfaz de usuario gráfica, Aplicación

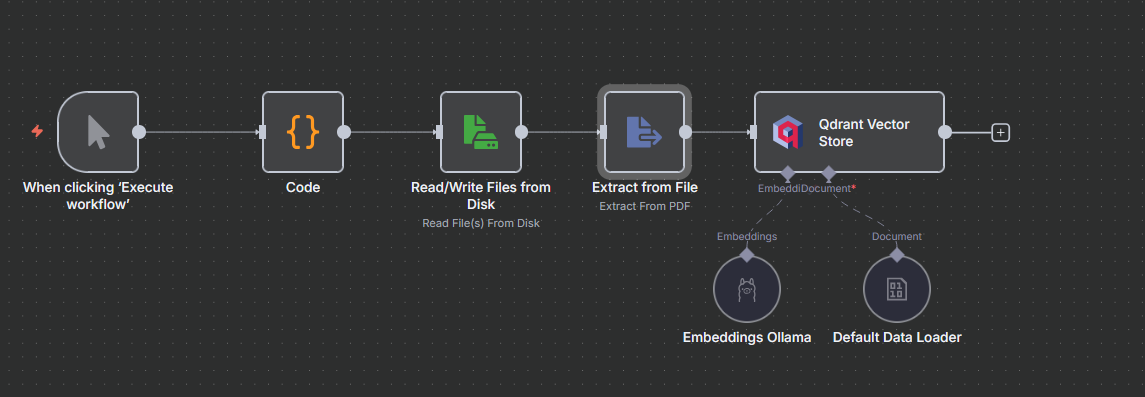
El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Para el embedding usamos ollama, porque lo podemos usar de local y es de las pocas opciones gratuitas para embedding.

Flujo completo:



**Flujo para agente inteligente**

Usamos un nodo “When chat message received “ para hablar con el agente

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Luego creamos el ai agent

Escala de tiempo

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

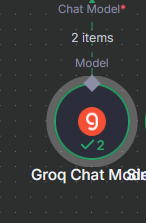
Usamos este prompt: Responde a la pregunta del usuario usando únicamente la información que obtengas de las herramientas disponibles.

Si no encuentras información suficiente, responde "No encontré información en los documentos".

Pregunta del usuario: {{ $json.chatInput }}.

Y le damos un máximo de 3 iteraciones para disminuir el tiempo de respuesta.

Usamos groq como chat model:



Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Para la memoria usamos un node de simple memory, ya que toda la información necesaria para responder se la dará el vector store, entonces no es necesario un nodo de memoria mas complejo:

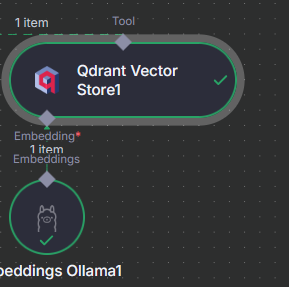
Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Ahora a el ai agent debemos conectarle otro nodo de vector store para que pueda acceder a la información que esta en el vector store:



Interfaz de usuario gráfica, Texto, Aplicación

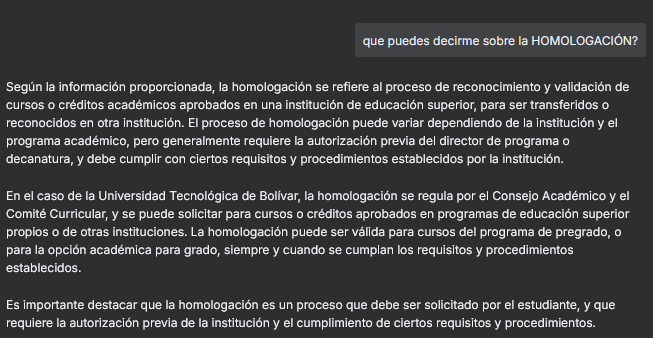
El contenido generado por IA puede ser incorrecto.

Nodo de ai agent completo:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Probando el ai agent:



Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.