Prueba técnica n8n - Documentación
Sistema RAG con n8n - PROINGES S.A.S.

Índice

Introducción

Este sistema implementa un **RAG (Retrieval-Augmented Generation)** para PROINGES S.A.S., permitiendo a los empleados consultar información contenida en documentos técnicos, procedimientos, manuales de ingeniería y registros corporativos a través de un bot de Telegram, sin necesidad de revisar manualmente cada documento.

¿Qué es un RAG?

Concepto

RAG (Retrieval-Augmented Generation) es una técnica de inteligencia artificial que combina dos capacidades fundamentales:

1. **Recuperación (Retrieval)**: Búsqueda de información relevante en una base de conocimiento
2. **Generación (Generation)**: Creación de respuestas coherentes utilizando modelos de lenguaje

¿Cómo funciona?

Usuario → Pregunta → Búsqueda en Vectores → Contexto Relevante → LLM → Respuesta

En lugar de que el modelo de IA responda únicamente basándose en su entrenamiento, el RAG:

- Busca documentos relevantes en una base de datos
- Extrae fragmentos relacionados con la consulta
- Utiliza estos fragmentos como contexto para generar respuestas precisas y fundamentadas

Ventajas

- **Respuestas basadas en documentos reales** de la empresa
- **Reducción de alucinaciones** del modelo de IA
- **Actualización dinámica** sin reentrenar modelos
- **Trazabilidad**: Se puede citar la fuente de cada respuesta
- **Privacidad**: Los datos permanecen en infraestructura controlada

Vectores y Búsqueda por Similitud

¿Qué son los Embeddings?

Los **embeddings** son representaciones numéricas (vectores) de texto que capturan su significado semántico. Palabras o frases con significados similares tienen vectores cercanos en el espacio matemático.

"motor eléctrico" → [0.23, 0.87, -0.45, ...]
"máquina eléctrica" → [0.25, 0.84, -0.43, ...] ← Vectores similares
"política de RR.HH." → [-0.67, 0.12, 0.89, ...] ← Vector distante

Búsqueda por Similitud Vectorial

En lugar de búsqueda tradicional por palabras clave, la búsqueda vectorial:

1. **Convierte la pregunta del usuario** en un vector usando el mismo modelo de embeddings
2. **Calcula distancias** entre el vector de la pregunta y todos los vectores en la base de datos
3. **Retorna los fragmentos más cercanos**, incluso si usan palabras diferentes

Ventaja clave: Encuentra información relevante aunque el usuario use sinónimos o parafrasee.

"¿Cuál es el procedimiento de mantenimiento preventivo?"
↓ (embedding)
[0.45, 0.78, -0.23, ...]
↓ (búsqueda por similitud)
Documento encontrado: "Protocolo de inspección periódica de equipos"
Vector: [0.47, 0.76, -0.25, ...] ← Alta similitud (90%)

Chunking: División Inteligente de Documentos

Para optimizar la búsqueda, los documentos se dividen en fragmentos (chunks):

- **Chunk Size: 100 tokens** - Tamaño de cada fragmento
- **Overlap: 200-250 tokens** - Superposición entre fragmentos

¿Por qué overlap?

El solapamiento evita que información importante se "corte" entre dos fragmentos:

Fragmento 1: "...el procedimiento requiere verificar temperatura..."
↓ OVERLAP ↓
Fragmento 2: "...verificar temperatura y presión antes de iniciar..."

Esto garantiza que el contexto no se pierda en los límites entre chunks.

Arquitectura de la Solución

El sistema se divide en **4 módulos principales**:

Tecnologías Utilizadas

Plataforma Principal

- **n8n** - Plataforma de automatización workflow

Almacenamiento y Base de Datos

- **Google Drive** - Repositorio de documentos fuente
- **PostgreSQL (Neon Console)** - Base de datos principal
- **pgvector** - Extensión para almacenamiento de vectores

Modelos de IA

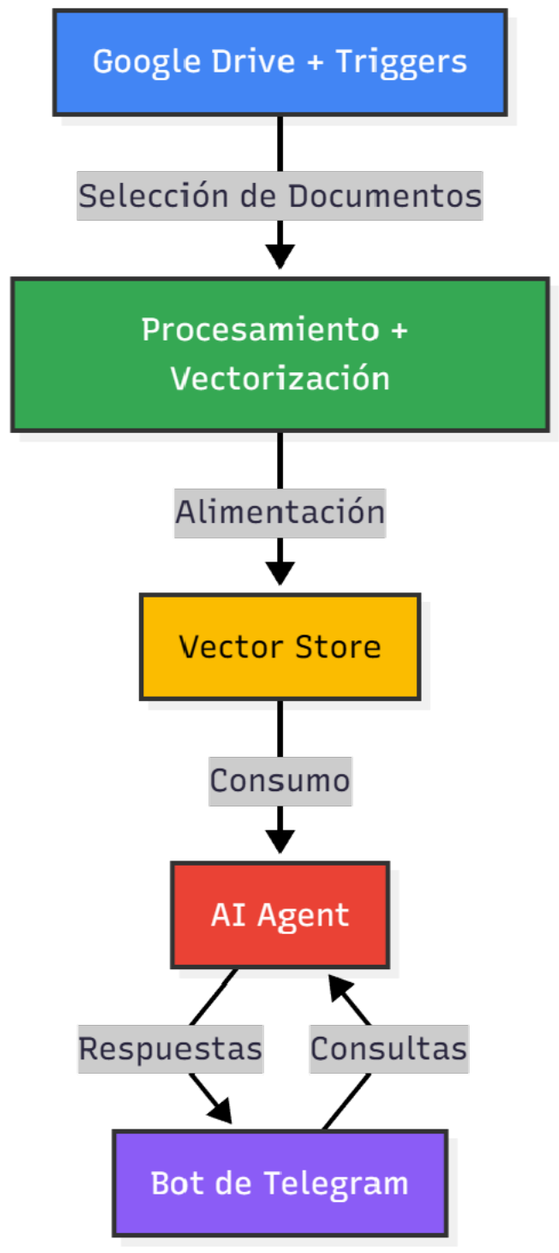
- **OpenAI GPT-4.1-mini** - Modelo de lenguaje para generación
- de respuestas
- **OpenAI text-embedding-3-large** - Modelo de embeddings
- para vectorización

Comunicación

- **Telegram Bot API** - Interface de usuario

Lenguajes y Herramientas

- **JavaScript** - Lógica personalizada en nodos Code
- **SQL** - Consultas a PostgreSQL



Base de Datos y Vectorización

Esquema de Base de Datos

Tabla: n8n_vectors (Almacenamiento de Vectores)

Esta tabla almacena los fragmentos de texto vectorizados de los documentos:

```
CREATE TABLE public.n8n_vectors (
    id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
    text text,                  -- Contenido del fragmento de texto
    metadata jsonb,             -- Metadatos (ubicación, líneas, etc.)
    embedding vector,           -- Vector matemático del fragmento
    file_id text,               -- Referencia al documento origen
    CONSTRAINT n8n_vectors_pkey PRIMARY KEY (id)
);

CREATE UNIQUE INDEX n8n_vectors_pkey ON public.n8n_vectors
USING btree (id);
```

Nota importante: El campo file_id debe agregarse manualmente, ya que n8n no lo crea por defecto.

Tabla: ingested_files (Control de Documentos)

Esta tabla mantiene registro de los documentos procesados para evitar duplicados y detectar cambios:

```
CREATE TABLE public.ingested_files (
    file_id text PRIMARY KEY,       -- ID único del archivo en Google Drive
    name text,                      -- Nombre del archivo
    mime_type text,                 -- Tipo MIME (pdf, docx, etc.)
    md5_checksum text,              -- Hash para detectar cambios
```

```
    last_ingested timestamp with time zone DEFAULT now(), -- Fecha de procesamiento

    CONSTRAINT ingested_files_pkey PRIMARY KEY (file_id)
);

CREATE UNIQUE INDEX ingested_files_pkey ON public.ingested_files
USING btree (file_id);
```

¿Por qué usar una base de datos vectorial?

Problema: Búsqueda Tradicional vs. Búsqueda Semántica

Búsqueda tradicional (SQL LIKE):

```
SELECT * FROM documents WHERE text LIKE '%motor%';
```

- Solo encuentra coincidencias exactas de palabras
- No entiende sinónimos ("motor" ≠ "máquina")
- No capta el contexto

Búsqueda vectorial (similitud):

```
SELECT * FROM n8n_vectors
ORDER BY embedding <-> query_vector
LIMIT 5;
```

- Encuentra información semánticamente similar
- Entiende sinónimos y paráfrasis
- Ordena por relevancia matemática

Ejemplo Práctico

Pregunta del usuario:

> "¿Cómo hago mantenimiento de la bomba hidrica?"

Con búsqueda tradicional:

- Buscaría "mantenimiento" Y "bomba"
- Podría perderse: "Procedimiento de inspección periódica de equipos de impulsión"

Con búsqueda vectorial:

- Convierte la pregunta a vector
- Encuentra documentos similares aunque usen palabras diferentes:
    - "Protocolo de revisión de bombas centrífugas"
    - "Inspección de sistemas de bombeo"
    - "Mantenimiento preventivo de equipos hidráulicos"

Relación entre Vectores y Documentos

Para mantener la trazabilidad y poder citar las fuentes, cada vector se relaciona con su documento de origen mediante el campo file_id:

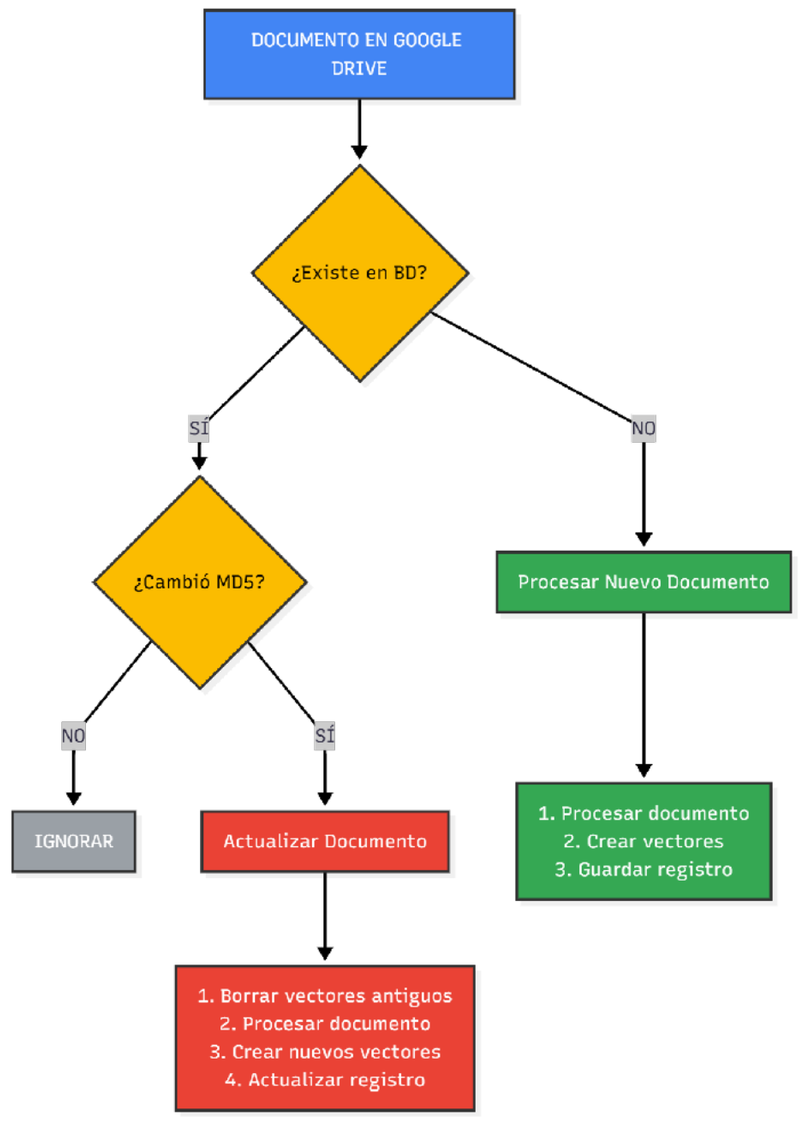
```
UPDATE n8n_vectors
SET file_id = '{{ $("Extraer variables").item.json.id }}'
WHERE metadata->'loc'->'lines' = '{{ $json.metadata.loc.lines.toJsonString() }}'
  AND file_id IS NULL;
```

Esta consulta:

1. Identifica vectores recién creados (file_id IS NULL)
2. Los vincula con el documento procesado usando los metadatos de ubicación
3. Permite luego recuperar el nombre del documento origen:

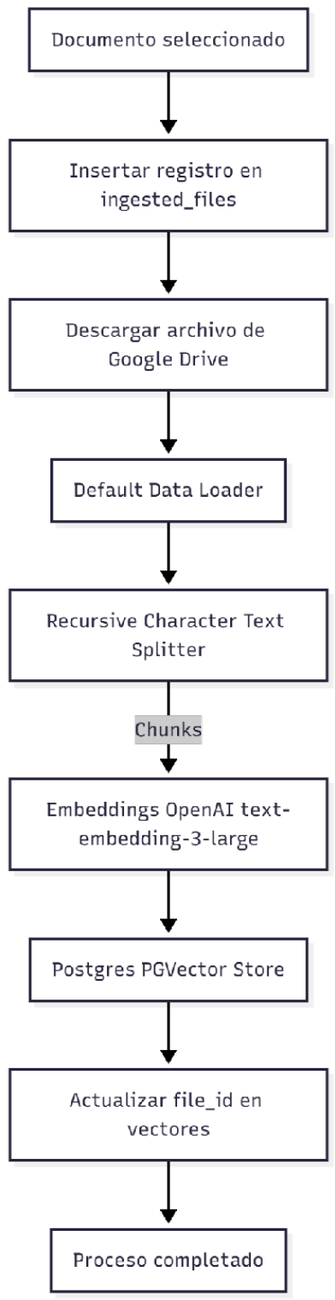
Diagramas de Flujo

Módulo 1: Selección de Documentos



Ventajas:

- **Eficiencia**: Solo procesa documentos nuevos o modificados
- **Actualización automática**: Detecta cambios mediante MD5 checksum
- **Limpieza**: Elimina vectores obsoletos antes de reprocesar



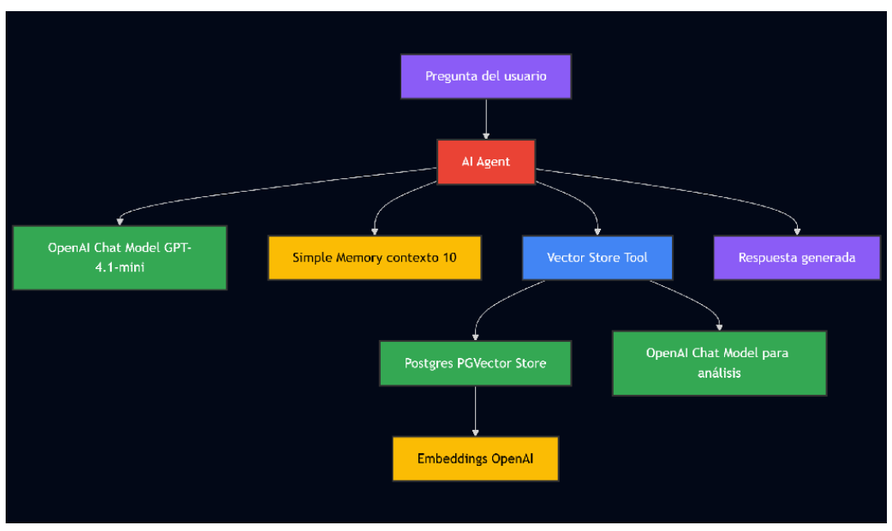
Descripción:

1. **Inserción de registro**: Guarda metadatos del documento
2. **Descarga**: Obtiene el archivo binario de Google Drive
3. **Carga de datos**: Extrae el texto del documento
4. **Chunking**: Divide en fragmentos con overlap (
4. 100 tokens, overlap 200-250)
5. **Vectorización**: Convierte cada chunk en embedding usando OpenAI
6. **Almacenamiento**: Guarda vectores en PostgreSQL con pgvector
7. **Vinculación**: Relaciona vectores con el documento origen

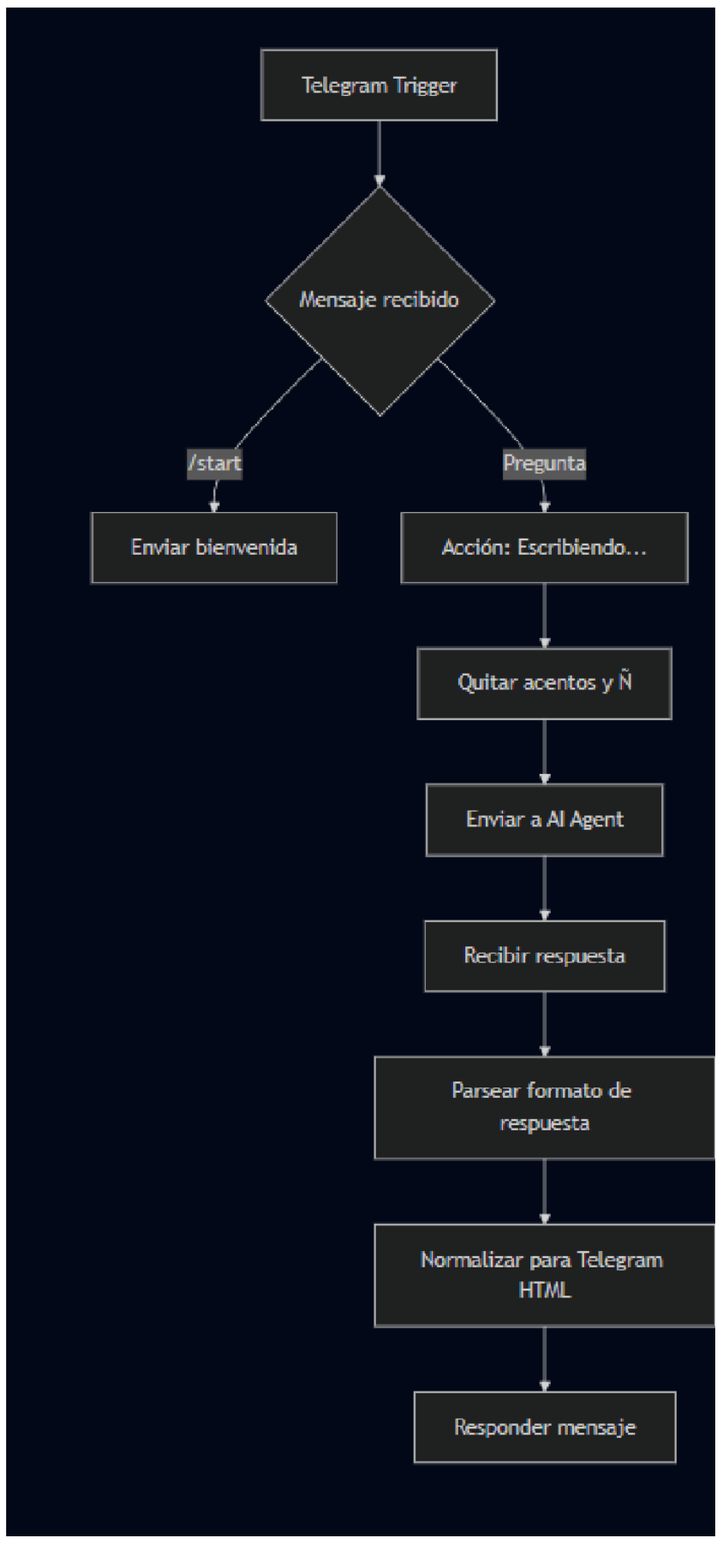
Parámetros de chunking:

- chunkSize: 100 - Tamaño óptimo para mantener contexto
- chunkOverlap: 200-250 - Superposición para evitar pérdida de información

Módulo 3: Consumo de Base de Datos (AI Agent)



Módulo 4: Conexión con Telegram



Descripción:

Componentes:

1. **Telegram Trigger**:
    - Escucha mensajes entrantes
    - Captura chat.id para identificar usuario
2. **Switch** (Enrutador):
    - /start → Mensaje de bienvenida
    - Cualquier otro texto → Procesamiento de pregunta
3. **Normalización de entrada**:

3. // Limpia formato Markdown para Telegram
3. text = text.replace(/**/g, ''); // Elimina negritas
3. text = text.replace(/*/g, ''); // Elimina asteriscos
3. text = text.replace(/__/g, ''); // Elimina cursivas
3. text = text.replace(/<[^>]*>/g, ''); // Limpia HTML

Configuración e Instalación

Requisitos Previos

1. **Cuenta n8n** (cloud o self-hosted)
2. **Base de datos PostgreSQL** con extensión pgvector (recomendado: Neon Console)
3. **Credenciales de servicios**:
    - OpenAI API Key
    - Google Drive OAuth2
    - Telegram Bot Token

Paso 1: Configurar PostgreSQL

1. Crear base de datos en Neon Console

1. Accede a Neon Console
2. Crea un nuevo proyecto
3. Obtén la cadena de conexión

CREATE EXTENSION IF NOT EXISTS vector;

1.3 Crear tablas

Tabla de vectores:

CREATE TABLE public.n8n_vectors (id uuid PRIMARY KEY DEFAULT gen_random_uuid(), text text, metadata jsonb, embedding vector, file_id text);CREATE UNIQUE INDEX n8n_vectors_pkey ON public.n8n_vectors USING btree (id);

CREATE TABLE public.ingested_files (file_id text PRIMARY KEY, name text, mime_type text, md5_checksum text, modified_time timestamp with time zone, last_ingested timestamp with time zone DEFAULT now()); CREATE UNIQUE INDEX ingested_files_pkey ON public.ingested_files USING btree (file_id); ```

Paso 2: Configurar Credenciales en n8n

2.1 OpenAI API

1. En n8n, ve a Settings → Credentials
2. Clic en Add Credential → OpenAI API
3. Ingresa tu API Key de OpenAI Platform
4. Nombre sugerido: OpenAi account

2.2 Google Drive OAuth2

1. Crea un proyecto en Google Cloud Console
2. Habilita Google Drive API
3. Crea credenciales OAuth 2.0
4. En n8n: Add Credential → Google Drive OAuth2 API
5. Completa Client ID, Client Secret
6. Autoriza el acceso

2.3 Telegram Bot

1. Habla con @BotFather en Telegram
2. Ejecuta /newbot y sigue instrucciones
3. Obtén el token del bot
4. En n8n: Add Credential → Telegram API
5. Pega el token

2.4 PostgreSQL Neon

1. En n8n: Add Credential → Postgres
2. Completa los datos de conexión de Neon: Host, Port 5432, Database, User, Password SSL: Enabled

Paso 3: Preparar Google Drive

1. Crea una carpeta para los documentos de la empresa
2. Copia el ID de la carpeta de la URL
3. Sube documentos en formatos compatibles PDF DOCX TXT

Instrucciones de Ejecución

Importar el Workflow a n8n

Opción 1: Importar desde JSON

1. Descargar el archivo: Guarda el JSON del workflow proporcionado
2. En n8n Web: Clic en el menú principal esquina superior izquierda Selecciona Import from file Carga el archivo RAG-PRUEBA-TECNICA.json
3. El workflow se importará con todos los nodos

Opción 2: Importar desde URL si aplica

1. Clic en Import from URL
2. Pega la URL del workflow compartido
3. Clic en Import

Configurar el Workflow

1. Nodos de Google Drive Nodo: Buscar archivos / directorios Selecciona credencial: Google Drive account En Folder ID, pega el ID de tu carpeta de Drive

Nodo: Descargar un archivo Selecciona la misma credencial de Google Drive

2. Nodos de PostgreSQL Conecta los siguientes nodos a tu credencial de Postgres: Obtener carga previa, Borrar referencias, Insertar documento, actualizar file_id en vectores, Postgres PGVector Store, Postgres PGVector Store1

Para cada uno: Clic en el nodo En Credential for Postgres, selecciona Postgres neon.com

3. Nodos de OpenAI Conecta los siguientes nodos a tu credencial de OpenAI: Embeddings OpenAI, OpenAI Chat Model, OpenAI Chat Model1

Para cada uno: Clic en el nodo En Credential for OpenAI, selecciona OpenAi account Verifica que el modelo sea: Embeddings: text-embedding-3-large Chat: gpt-4.1-mini

4. Nodos de Telegram Conecta los siguientes nodos a tu credencial de Telegram: Telegram Trigger, Bienvenida, Escribiendo, Responder mensajes

Para cada uno: Clic en el nodo En Credential for Telegram, selecciona Telegram account

Ajustar Parámetros del Sistema

AI Agent Nodo principal Clic en el nodo AI Agent Revisa el System Message que define el comportamiento del asistente Puedes personalizar: Tono de respuesta, Formato de referencias, Nivel de detalle
Chunking División de documentos Nodo: Recursive Character Text Splitter Parámetros actuales: chunkSize 100, chunkOverlap 200

Activar y Ejecutar

Primera Ejecución: Cargar Documentos

1. **Desactivar el trigger automático** (opcional)
   - Clic derecho en Schedule Trigger
   - Selecciona **Disable**
2. **Ejecutar manualmente**:
   - Clic en el nodo When clicking 'Execute workflow'
   - Clic en **Execute Node** (arriba a la derecha)
   - Observa el flujo de ejecución
3. **Verificar progreso**:
   - Los nodos se marcarán en verde al completarse
   - Revisa que los documentos se procesen correctamente
   - Verifica en PostgreSQL que las tablas se llenen:

Activar el Bot de Telegram

1. **Activar el workflow**:
   - Toggle **Active** en la esquina superior derecha
   - El nodo Telegram Trigger se activará automáticamente

2. **Probar el bot**:
   - Abre Telegram
   - Busca tu bot por su username
   - Envía /start
   - Deberías recibir el mensaje de bienvenida

3. **Hacer consultas**:
   Usuario: ¿Cuál es el procedimiento de soldadura GMAW?

Bot: [Respuesta basada en documentos...]

Mantenimiento y Actualizaciones

Actualización Automática de Documentos

Nodo: Schedule Trigger

El sistema revisa Google Drive automáticamente cada cierto tiempo:

```
// Configuración actual: Diariamente
rule: {
  interval: [{}]  // Cada día
}
```

Modificar frecuencia:

1. Clic en Schedule Trigger
2. Ajusta interval:
    - Cada hora
    - Cada 12 horas
    - Semanal
3. Guarda

Agregar Nuevos Documentos

1. **Simplemente súbelos a la carpeta de Google Drive**
2. El sistema detectará automáticamente:
    - Archivos nuevos
    - Archivos modificados (cambio en MD5)
3. Los procesará en la próxima ejecución del schedule

Monitoreo

Verificar ejecuciones:

- En n8n, ve a **Executions** (menú lateral)
- Revisa logs de ejecuciones recientes
- Identifica errores si los hay