**PROMPT:**

# Contexto  
Eres un asistente de investigación automatizado especializado en búsqueda y análisis de literatura científica.   
Tu función principal es identificar artículos académicos recientes sobre **el tema que el usuario te ha especificado junto al rango de fechas,** evaluar su relevancia y extraer hallazgos clave para alimentar una base de datos de seguimiento diario.  
El objetivo es reducir el tiempo invertido en la búsqueda manual de papers, proporcionando una lista filtrada y comentada de artículos altamente relevantes.  
---  
  
## Pasos  
### Nodo 1 – Búsqueda de artículos  
1. Realiza una búsqueda en bases de datos académicas o APIs científicas (como Semantic Scholar, Scopus, Web o Science, Scielo, Pubmed, Crossref, o similar).  
2. Usa la siguiente consulta base:  
> "Tema especificado por el usuario junto al rango de fechas"   
Filtra por: publicaciones del **último mes**, **revisadas por pares**, **en inglés o español**, y que incluyan **resumen disponible**.  
3. Devuelve una lista estructurada con:   
- Título del artículo   
- Autores   
- Año de publicación   
- URL o DOI   
- Resumen   
- Fuente o base de datos  
---  
### Nodo 2 – Iteración  
Itera sobre cada resultado devuelto, enviando cada uno individualmente al siguiente nodo (resumen y evaluación).  
---  
### Nodo 3 – Evaluación y resumen IA  
Para cada artículo recibido, aplica el siguiente \*prompt\*:  
> Evalúa la relevancia del siguiente artículo para una investigación sobre **el tema solicitado.**   
> 1. Responde con "Sí" si es altamente relevante, o "No" si no lo es.   
> 2. Resume brevemente el artículo (máx. 100 palabras).   
> 3. Extrae los **3 hallazgos o conclusiones más destacados**.   
> Devuelve tu respuesta en el siguiente formato:  
> - **Relevante:** [Sí/No]   
> - **Resumen:** [texto breve]   
> - **Hallazgos clave:**    
> 1. [hallazgo 1]   
> 2. [hallazgo 2]   
> 3. [hallazgo 3]  
---  
### Nodo 4 – Registro en hoja de cálculo  
Registra cada resultado en una hoja (Google Sheets o Airtable) con las columnas:  
- Título   
- URL/DOI   
- Relevante (Sí/No)   
- Resumen breve   
- Hallazgos clave   
---  
### Nodo 5 – Integración con gestor bibliográfico (opcional)  
Si la evaluación es **“Sí”**, envía los metadatos del artículo al gestor bibliográfico (Zotero o Mendeley) mediante su API para guardar la referencia en una colección predefinida (p. ej. “Tema especificado”).  
---  
### Nodo 6 – Notificación (opcional)  
Si la evaluación fue **“Sí”** y contiene términos como “significant impact”, “critical loss”, “biodiversity decline”, etc., envía una alerta vía **Slack** o **correo electrónico** al investigador con:  
- El título del artículo   
- Enlace   
- Hallazgos clave   
---  
## Lenguaje y Formato  
- **Lenguaje:** Técnico, académico y objetivo.   
- **Formato de salida:**    
Texto estructurado en listas o bloques con etiquetas claras (Título, Resumen, Hallazgos).   
Evita opiniones o juicios subjetivos.  
---  
## Ejemplo  
**Entrada:**    
> "Tema especificado por el usuario leído en Python y Streamlit – Una revisión global   
> Resumen del artículo…."  
**Salida esperada:**    
- **Relevante:** Sí   
- **Resumen:** Revisión global de los efectos del tema especificado por el usuario.   
- **Hallazgos clave:**    
1. Hallazgo clave 1.   
2. Hallazgo clave 2.   
3. Hallazgo clave31.  
---  
## Notas  
- Si la API académica no está disponible, puedes usar scrapers o servicios de búsqueda pública como Semantic Scholar, [Lens.org](https://web.telegram.org/a/Lens.org) o PubMed.   
- Ajusta la frecuencia del flujo (por ejemplo, semanal) según tus necesidades de actualización.   
- Si se detectan duplicados, omite entradas repetidas antes de registrar.   
- Todos los resultados deben mantenerse en un formato limpio y homogéneo para facilitar su revisión posterior. Para la automatización de n8n debes generar el workflow en formato json y debe ser consumida por una aplicación en Python con Streamlit, usando la base de datos Supabase y debe generar estadísticas descriptivas, gráficos explicativos y un reporte en pdf

**HERRAMIENTA: DEEPSEEK**

**PROMPT:**

> # Contexto Eres un asistente de investigación automatizado especializado en búsqueda y análisis de literatura científica. Tu función principal es identificar artículos académicos recientes sobre el tema que el usuario te ha especificado junto al rango de fechas, evaluar su relevancia y extraer hallazgos clave para alimentar una base de datos de seguimiento diario. El objetivo es reducir el tiempo invertido en la búsqueda manual de papers, proporcionando una lista filtrada y comentada de artículos altamente relevantes. --- ## Pasos ### Nodo 1 – Búsqueda de artículos 1. Realiza una búsqueda en bases de datos académicas o APIs científicas (como Semantic Scholar, Scopus, Web o Science, Scielo, Pubmed, Crossref, o similar). 2. Usa la siguiente consulta base: > "Tema especificado por el usuario junto al rango de fechas" Filtra por: publicaciones del último mes, revisadas por pares, en inglés o español, y que incluyan resumen disponible. 3. Devuelve una lista estructurada con: - Título del artículo - Autores - Año de publicación - URL o DOI - Resumen - Fuente o base de datos --- ### Nodo 2 – Iteración Itera sobre cada resultado devuelto, enviando cada uno individualmente al siguiente nodo (resumen y evaluación). --- ### Nodo 3 – Evaluación y resumen IA Para cada artículo recibido, aplica el siguiente \*prompt\*: > Evalúa la relevancia del siguiente artículo para una investigación sobre el tema solicitado. > 1. Responde con "Sí" si es altamente relevante, o "No" si no lo es. > 2. Resume brevemente el artículo (máx. 100 palabras). > 3. Extrae los 3 hallazgos o conclusiones más destacados. > Devuelve tu respuesta en el siguiente formato: > - Relevante: [Sí/No] > - Resumen: [texto breve] > - Hallazgos clave: > 1. [hallazgo 1] > 2. [hallazgo 2] > 3. [hallazgo 3] --- ### Nodo 4 – Registro en hoja de cálculo Registra cada resultado en una hoja (Google Sheets o Airtable) con las columnas: - Título - URL/DOI - Relevante (Sí/No) - Resumen breve - Hallazgos clave --- ### Nodo 5 – Integración con gestor bibliográfico (opcional) Si la evaluación es “Sí”, envía los metadatos del artículo al gestor bibliográfico (Zotero o Mendeley) mediante su API para guardar la referencia en una colección predefinida (p. ej. “Tema especificado”). --- ### Nodo 6 – Notificación (opcional) Si la evaluación fue “Sí” y contiene términos como “significant impact”, “critical loss”, “biodiversity decline”, etc., envía una alerta vía Slack o correo electrónico al investigador con: - El título del artículo - Enlace - Hallazgos clave --- ## Lenguaje y Formato - Lenguaje: Técnico, académico y objetivo. - Formato de salida: Texto estructurado en listas o bloques con etiquetas claras (Título, Resumen, Hallazgos). Evita opiniones o juicios subjetivos. --- ## Ejemplo Entrada: > "Tema especificado por el usuario leído en Python y Streamlit – Una revisión global > Resumen del artículo…." Salida esperada: - Relevante: Sí - Resumen: Revisión global de los efectos del tema especificado por el usuario. - Hallazgos clave: 1. Hallazgo clave 1. 2. Hallazgo clave 2. 3. Hallazgo clave31. --- ## Notas - Si la API académica no está disponible, puedes usar scrapers o servicios de búsqueda pública como Semantic Scholar, Lens.org o PubMed. - Ajusta la frecuencia del flujo (por ejemplo, semanal) según tus necesidades de actualización. - Si se detectan duplicados, omite entradas repetidas antes de registrar. - Todos los resultados deben mantenerse en un formato limpio y homogéneo para facilitar su revisión posterior. Para la automatización de n8n debes generar el workflow en formato json y debe ser consumida por una aplicación en Python con Streamlit, usando la base de datos Supabase y debe generar estadísticas descriptivas, gráficos explicativos y un reporte en pdf

**SALIDA:**

Verifica que el tema de búsqueda y el rango de fechas sea ingresado por el usuario en la aplicación de Python con Streamlit, para ser enviada como parámetros al flujo automatizado en n8n, para luego este devuelva los resultados a la aplicaci{on de Python con Streamlit y a partir de los datos grabados en Supabase genere las estadísticas descriptivas, la elaboración de gráficos explicativos y genere estos resultados en un reporte en pdf. Actualiza todo el flujo de n8n y la aplicaci{on en Python con Streamlit completamente funcional

**Sistema Completo Actualizado: Streamlit + n8n + Supabase**

**1. Aplicación Streamlit Actualizada**

python

# app.py

import streamlit as st

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from supabase import create\_client

from fpdf import FPDF

import plotly.express as px

import plotly.graph\_objects as go

from datetime import datetime, timedelta

import requests

import json

import time

# Configuración de página

st.set\_page\_config(

page\_title="Scientific Literature Monitor",

page\_icon="📊",

layout="wide"

)

# Clases para reportes PDF

class PDFReport(FPDF):

def header(self):

self.set\_font('Arial', 'B', 15)

self.cell(0, 10, 'Reporte de Literatura Científica', 0, 1, 'C')

self.ln(10)

def chapter\_title(self, title):

self.set\_font('Arial', 'B', 12)

self.cell(0, 10, title, 0, 1, 'L')

self.ln(4)

def chapter\_body(self, body):

self.set\_font('Arial', '', 10)

self.multi\_cell(0, 8, body)

self.ln()

# Cliente Supabase

@st.cache\_resource

def init\_supabase():

return create\_client(

st.secrets["SUPABASE\_URL"],

st.secrets["SUPABASE\_KEY"]

)

# Funciones principales

def trigger\_n8n\_workflow(topic, start\_date, end\_date):

"""Dispara el workflow de n8n con los parámetros del usuario"""

n8n\_webhook\_url = st.secrets["N8N\_WEBHOOK\_URL"]

payload = {

"topic": topic,

"start\_date": start\_date.isoformat(),

"end\_date": end\_date.isoformat(),

"execution\_id": f"streamlit\_{int(time.time())}"

}

try:

response = requests.post(n8n\_webhook\_url, json=payload, timeout=30)

if response.status\_code == 200:

return True, "Workflow iniciado correctamente"

else:

return False, f"Error en n8n: {response.status\_code}"

except Exception as e:

return False, f"Error conectando con n8n: {str(e)}"

def fetch\_papers\_from\_supabase(topic\_filter=None, date\_range=None):

"""Obtiene papers de Supabase con filtros"""

supabase = init\_supabase()

query = supabase.table('papers').select('\*').order('created\_at', desc=True)

if topic\_filter:

query = query.ilike('title', f'%{topic\_filter}%')

if date\_range:

start\_date, end\_date = date\_range

query = query.gte('created\_at', start\_date.isoformat()).lte('created\_at', end\_date.isoformat())

response = query.execute()

return pd.DataFrame(response.data) if response.data else pd.DataFrame()

def generate\_descriptive\_stats(df):

"""Genera estadísticas descriptivas"""

if df.empty:

return {}

stats = {

'total\_papers': len(df),

'relevant\_papers': len(df[df['relevant'] == 'Sí']),

'relevance\_rate': (len(df[df['relevant'] == 'Sí']) / len(df)) \* 100 if len(df) > 0 else 0,

'papers\_by\_year': df['publication\_year'].value\_counts().to\_dict(),

'recent\_papers': len(df[df['created\_at'] >= (datetime.now() - timedelta(days=7)).isoformat()])

}

return stats

def create\_visualizations(df, stats):

"""Crea visualizaciones interactivas"""

if df.empty:

return None, None, None

# Gráfico de relevancia

relevancy\_data = df['relevant'].value\_counts()

fig1 = px.pie(

values=relevancy\_data.values,

names=relevancy\_data.index,

title='Distribución por Relevancia',

color=relevancy\_data.index,

color\_discrete\_map={'Sí': '#00CC96', 'No': '#EF553B'}

)

# Gráfico temporal de publicaciones

if stats.get('papers\_by\_year'):

yearly\_data = pd.DataFrame({

'Año': list(stats['papers\_by\_year'].keys()),

'Cantidad': list(stats['papers\_by\_year'].values())

}).sort\_values('Año')

fig2 = px.bar(yearly\_data, x='Año', y='Cantidad',

title='Publicaciones por Año',

color='Cantidad',

color\_continuous\_scale='Viridis')

else:

fig2 = None

# Gráfico de evolución temporal de procesamiento

df['created\_date'] = pd.to\_datetime(df['created\_at']).dt.date

daily\_counts = df.groupby('created\_date').size().reset\_index(name='count')

fig3 = px.line(daily\_counts, x='created\_date', y='count',

title='Artículos Procesados por Día',

markers=True)

return fig1, fig2, fig3

def generate\_pdf\_report(df, stats, topic, date\_range):

"""Genera reporte PDF completo"""

pdf = PDFReport()

pdf.add\_page()

# Encabezado

pdf.set\_font('Arial', 'B', 16)

pdf.cell(0, 10, 'REPORTE DE LITERATURA CIENTÍFICA', 0, 1, 'C')

pdf.ln(5)

# Información del análisis

pdf.chapter\_title('INFORMACIÓN DEL ANÁLISIS')

pdf.chapter\_body(f"""

Tema de investigación: {topic}

Período analizado: {date\_range[0].strftime('%Y-%m-%d')} a {date\_range[1].strftime('%Y-%m-%d')}

Fecha de generación: {datetime.now().strftime('%Y-%m-%d %H:%M')}

""")

# Resumen ejecutivo

pdf.chapter\_title('RESUMEN EJECUTIVO')

pdf.chapter\_body(f"""

Total de artículos analizados: {stats.get('total\_papers', 0)}

Artículos relevantes identificados: {stats.get('relevant\_papers', 0)}

Tasa de relevancia: {stats.get('relevance\_rate', 0):.2f}%

Artículos procesados en la última semana: {stats.get('recent\_papers', 0)}

""")

# Hallazgos clave de artículos relevantes

relevant\_papers = df[df['relevant'] == 'Sí']

if not relevant\_papers.empty:

pdf.chapter\_title('HALLAZGOS CLAVE - ARTÍCULOS RELEVANTES')

for idx, row in relevant\_papers.head(10).iterrows():

pdf.set\_font('Arial', 'B', 10)

pdf.cell(0, 8, f"Título: {row.get('title', 'N/A')}", 0, 1)

pdf.set\_font('Arial', '', 9)

authors = row.get('authors', [])

if authors and isinstance(authors, list):

author\_names = [author.get('name', '') for author in authors if author.get('name')]

pdf.cell(0, 6, f"Autores: {', '.join(author\_names)}", 0, 1)

pdf.cell(0, 6, f"Año: {row.get('publication\_year', 'N/A')}", 0, 1)

key\_findings = row.get('key\_findings', '')

if key\_findings and len(key\_findings) > 50:

pdf.multi\_cell(0, 6, f"Hallazgos: {key\_findings[:200]}...")

else:

pdf.multi\_cell(0, 6, f"Hallazgos: {key\_findings}")

pdf.ln(2)

# Distribución por año

if stats.get('papers\_by\_year'):

pdf.chapter\_title('DISTRIBUCIÓN POR AÑO DE PUBLICACIÓN')

for year, count in sorted(stats['papers\_by\_year'].items()):

pdf.cell(0, 8, f"Año {year}: {count} artículos", 0, 1)

return pdf.output(dest='S').encode('latin-1')

# Interfaz principal

def main():

st.title("📊 Sistema de Monitorización de Literatura Científica")

st.markdown("---")

# Sidebar para configuración

with st.sidebar:

st.header("🔍 Configuración de Búsqueda")

# Entrada del usuario

research\_topic = st.text\_input(

"Tema de investigación:",

placeholder="Ej: machine learning, climate change, neural networks...",

help="Ingresa el tema específico que quieres investigar"

)

col1, col2 = st.columns(2)

with col1:

start\_date = st.date\_input(

"Fecha inicio:",

value=datetime.now() - timedelta(days=30)

)

with col2:

end\_date = st.date\_input(

"Fecha fin:",

value=datetime.now()

)

# Validación de fechas

if start\_date > end\_date:

st.error("❌ La fecha de inicio debe ser anterior a la fecha fin")

return

# Botón para ejecutar búsqueda

if st.button("🚀 Ejecutar Búsqueda Automatizada", type="primary", use\_container\_width=True):

if research\_topic.strip():

with st.spinner("Iniciando búsqueda automatizada..."):

success, message = trigger\_n8n\_workflow(research\_topic, start\_date, end\_date)

if success:

st.success("✅ " + message)

st.info("⏳ Los resultados estarán disponibles en unos minutos. Actualiza la página.")

else:

st.error("❌ " + message)

else:

st.warning("⚠️ Por favor ingresa un tema de investigación")

st.markdown("---")

st.header("📂 Filtros de Visualización")

topic\_filter = st.text\_input("Filtrar por palabra clave:", placeholder="machine learning...")

visualization\_range = st.date\_input(

"Rango para visualización:",

value=(datetime.now() - timedelta(days=30), datetime.now()),

key="viz\_range"

)

# Contenido principal

tab1, tab2, tab3 = st.tabs(["📈 Dashboard", "📋 Artículos", "📊 Reportes"])

with tab1:

st.header("Dashboard de Métricas")

# Cargar datos

date\_range\_tuple = (visualization\_range[0], visualization\_range[1]) if isinstance(visualization\_range, tuple) else (visualization\_range, visualization\_range)

df = fetch\_papers\_from\_supabase(topic\_filter, date\_range\_tuple)

if df.empty:

st.info("💡 No hay datos disponibles. Ejecuta una búsqueda primero o ajusta los filtros.")

else:

# Métricas principales

stats = generate\_descriptive\_stats(df)

col1, col2, col3, col4 = st.columns(4)

with col1:

st.metric("Total de Artículos", stats.get('total\_papers', 0))

with col2:

st.metric("Artículos Relevantes", stats.get('relevant\_papers', 0))

with col3:

st.metric("Tasa de Relevancia", f"{stats.get('relevance\_rate', 0):.1f}%")

with col4:

st.metric("Recientes (7 días)", stats.get('recent\_papers', 0))

# Visualizaciones

fig1, fig2, fig3 = create\_visualizations(df, stats)

if fig1 and fig2 and fig3:

col1, col2 = st.columns(2)

with col1:

st.plotly\_chart(fig1, use\_container\_width=True)

with col2:

st.plotly\_chart(fig2, use\_container\_width=True)

st.plotly\_chart(fig3, use\_container\_width=True)

with tab2:

st.header("Lista de Artículos")

if not df.empty:

# Filtros adicionales

col1, col2 = st.columns(2)

with col1:

relevance\_filter = st.selectbox("Filtrar por relevancia:", ["Todos", "Sí", "No"])

with col2:

sort\_by = st.selectbox("Ordenar por:", ["Fecha creación", "Año publicación", "Relevancia"])

# Aplicar filtros

filtered\_df = df.copy()

if relevance\_filter != "Todos":

filtered\_df = filtered\_df[filtered\_df['relevant'] == relevance\_filter]

# Ordenar

if sort\_by == "Fecha creación":

filtered\_df = filtered\_df.sort\_values('created\_at', ascending=False)

elif sort\_by == "Año publicación":

filtered\_df = filtered\_df.sort\_values('publication\_year', ascending=False)

elif sort\_by == "Relevancia":

filtered\_df = filtered\_df.sort\_values('relevant', ascending=False)

# Mostrar tabla

display\_columns = ['title', 'authors', 'publication\_year', 'relevant', 'created\_at']

display\_df = filtered\_df[display\_columns].copy()

# Formatear autores para visualización

def format\_authors(authors\_list):

if isinstance(authors\_list, list):

names = [author.get('name', '') for author in authors\_list[:3] if author.get('name')]

return ', '.join(names) + ('...' if len(authors\_list) > 3 else '')

return 'N/A'

display\_df['authors'] = display\_df['authors'].apply(format\_authors)

display\_df['created\_at'] = pd.to\_datetime(display\_df['created\_at']).dt.strftime('%Y-%m-%d')

st.dataframe(

display\_df,

column\_config={

"title": "Título",

"authors": "Autores",

"publication\_year": "Año",

"relevant": "Relevante",

"created\_at": "Procesado"

},

use\_container\_width=True,

height=400

)

st.caption(f"Mostrando {len(filtered\_df)} de {len(df)} artículos")

else:

st.info("💡 No hay artículos para mostrar con los filtros actuales.")

with tab3:

st.header("Generar Reportes")

if not df.empty:

col1, col2 = st.columns([2, 1])

with col1:

st.subheader("Reporte PDF Personalizado")

report\_topic = st.text\_input("Título del reporte:", value=research\_topic or "Análisis de Literatura")

if st.button("📄 Generar Reporte PDF", type="primary"):

with st.spinner("Generando reporte PDF..."):

pdf\_data = generate\_pdf\_report(df, stats, report\_topic, date\_range\_tuple)

st.download\_button(

label="⬇️ Descargar Reporte PDF",

data=pdf\_data,

file\_name=f"reporte\_cientifico\_{datetime.now().strftime('%Y%m%d\_%H%M')}.pdf",

mime="application/pdf",

use\_container\_width=True

)

with col2:

st.subheader("Exportar Datos")

# Exportar a CSV

csv\_data = df.to\_csv(index=False).encode('utf-8')

st.download\_button(

label="📊 Exportar CSV",

data=csv\_data,

file\_name=f"datos\_cientificos\_{datetime.now().strftime('%Y%m%d')}.csv",

mime="text/csv",

use\_container\_width=True

)

# Exportar a JSON

json\_data = df.to\_json(orient='records', indent=2)

st.download\_button(

label="🔤 Exportar JSON",

data=json\_data,

file\_name=f"datos\_cientificos\_{datetime.now().strftime('%Y%m%d')}.json",

mime="application/json",

use\_container\_width=True

)

else:

st.info("💡 No hay datos disponibles para generar reportes.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

## 2. Workflow n8n Actualizado (JSON Completo)

Json

{

"name": "Scientific Literature Monitor - Enhanced",

"nodes": [

{

"parameters": {

"path": "streamlit-webhook",

"responseMode": "responseNode",

"responseData": "firstEntry"

},

"id": "streamlit-webhook",

"name": "Streamlit Webhook",

"type": "n8n-nodes-base.webhook",

"typeVersion": 1,

"position": [240, 300],

"webhookId": "streamlit-trigger"

},

{

"parameters": {

"functionCode": "// Prepara parámetros para Semantic Scholar\nconst { topic, start\_date, end\_date, execution\_id } = $input.first().json;\n\n// Calcular rango de años\nconst startYear = new Date(start\_date).getFullYear();\nconst endYear = new Date(end\_date).getFullYear();\n\nreturn [{\n topic: topic,\n start\_year: startYear,\n end\_year: endYear,\n execution\_id: execution\_id,\n original\_payload: $input.first().json\n}];"

},

"id": "process-parameters",

"name": "Process Parameters",

"type": "n8n-nodes-base.function",

"typeVersion": 1,

"position": [460, 300]

},

{

"parameters": {

"method": "GET",

"url": "https://api.semanticscholar.org/graph/v1/paper/search",

"authentication": "none",

"queryParameters": {

"parameters": [

{

"name": "query",

"value": "={{ $json.topic }}"

},

{

"name": "fields",

"value": "title,authors,year,abstract,url,doi,venue,publicationTypes,citationCount"

},

{

"name": "year",

"value": "={{ $json.start\_year + '-' + $json.end\_year }}"

},

{

"name": "limit",

"value": "50"

},

{

"name": "openAccessPdf",

"value": ""

}

]

},

"sendHeaders": true,

"headerParameters": {

"parameters": [

{

"name": "User-Agent",

"value": "n8n-research-bot/2.0"

}

]

},

"timeout": 60000

},

"id": "semantic-scholar-api",

"name": "Semantic Scholar API",

"type": "n8n-nodes-base.httpRequest",

"typeVersion": 4,

"position": [680, 300]

},

{

"parameters": {

"conditions": {

"options": {

"caseSensitive": true

},

"conditions": [

{

"id": "data-check",

"leftValue": "={{ $json.data }}",

"rightValue": "null",

"operator": {

"type": "array",

"operation": "notEmpty"

}

}

]

}

},

"id": "check-results",

"name": "Check API Results",

"type": "n8n-nodes-base.if",

"typeVersion": 2,

"position": [900, 300]

},

{

"parameters": {

"batchSize": 1,

"options": {}

},

"id": "split-in-batches",

"name": "Split Articles",

"type": "n8n-nodes-base.splitInBatches",

"typeVersion": 1,

"position": [1120, 200]

},

{

"parameters": {

"model": "gpt-4",

"prompt": "Evalúa la relevancia del siguiente artículo para investigación sobre '{{ $node('Process Parameters').json.topic }}'.\n\nINFORMACIÓN DEL ARTÍCULO:\nTítulo: {{ $json.title }}\nAño: {{ $json.year }}\nResumen: {{ $json.abstract || 'No disponible' }}\nAutores: {{ $json.authors.map(author => author.name).join(', ') }}\nVenue: {{ $json.venue || 'No disponible' }}\n\nINSTRUCCIONES:\n1. Responde EXACTAMENTE 'Sí' si es altamente relevante para el tema, 'No' si no lo es\n2. Proporciona un resumen breve (máximo 100 palabras) en español\n3. Extrae los 3 hallazgos o conclusiones más destacados\n\nFORMATO DE RESPUESTA OBLIGATORIO:\n- Relevante: [Sí/No]\n- Resumen: [texto breve aquí]\n- Hallazgos clave:\n1. [primer hallazgo]\n2. [segundo hallazgo]\n3. [tercer hallazgo]",

"options": {

"temperature": 0.1,

"maxTokens": 600

}

},

"id": "openai-evaluation",

"name": "AI Relevance Evaluation",

"type": "n8n-nodes-base.openAi",

"typeVersion": 3,

"position": [1340, 200],

"credentials": {

"openAiApi": {

"id": "openai-api-credential"

}

}

},

{

"parameters": {

"operation": "insert",

"table": "papers",

"data": {

"title": "={{ $json.title }}",

"authors": "={{ $json.authors }}",

"publication\_year": "={{ $json.year }}",

"doi": "={{ $json.doi }}",

"url": "={{ $json.url }}",

"abstract": "={{ $json.abstract }}",

"venue": "={{ $json.venue }}",

"citation\_count": "={{ $json.citationCount }}",

"relevant": "={{ $node('AI Relevance Evaluation').json.answer.split('Relevante: ')[1]?.split('\\\\n')[0]?.trim() }}",

"summary": "={{ $node('AI Relevance Evaluation').json.answer.split('Resumen: ')[1]?.split('\\\\n')[0]?.trim() }}",

"key\_findings": "={{ $node('AI Relevance Evaluation').json.answer.split('Hallazgos clave:')[1]?.trim() }}",

"execution\_id": "={{ $node('Process Parameters').json.execution\_id }}",

"search\_topic": "={{ $node('Process Parameters').json.topic }}",

"processed\_at": "={{ new Date().toISOString() }}"

},

"additionalFields": {},

"returnFields": ["id", "title", "relevant"]

},

"id": "supabase-storage",

"name": "Supabase Storage",

"type": "n8n-nodes-base.supabase",

"typeVersion": 1,

"position": [1560, 200],

"credentials": {

"supabaseApi": {

"id": "supabase-credential"

}

}

},

{

"parameters": {

"message": "✅ Búsqueda completada para: {{ $node('Process Parameters').json.topic }}\n📊 Artículos procesados: {{ $json.data.length }}\n🎯 Relevantes identificados: {{ $node('Check Relevant Papers').json.relevantCount }}\n⏰ Tiempo total: {{ $now.diff($node('Process Parameters').json.original\_payload.execution\_id.split('\_')[1] \* 1000, 'minutes') }} minutos",

"otherOptions": {}

},

"id": "success-response",

"name": "Success Response",

"type": "n8n-nodes-base.webhook",

"typeVersion": 1,

"position": [1780, 300]

},

{

"parameters": {

"message": "❌ No se encontraron artículos para: {{ $node('Process Parameters').json.topic }}\n📅 Rango: {{ $node('Process Parameters').json.start\_year }}-{{ $node('Process Parameters').json.end\_year }}\n💡 Sugerencia: Intenta con términos más amplios o diferente rango temporal"

},

"id": "no-results-response",

"name": "No Results Response",

"type": "n8n-nodes-base.webhook",

"typeVersion": 1,

"position": [1780, 500]

},

{

"parameters": {

"functionCode": "// Contar artículos relevantes\nconst relevantCount = $input.all().filter(item => item.json.relevant === 'Sí').length;\nreturn { relevantCount };"

},

"id": "check-relevant-papers",

"name": "Check Relevant Papers",

"type": "n8n-nodes-base.function",

"typeVersion": 1,

"position": [1560, 400]

},

{

"parameters": {

"conditions": {

"options": {

"caseSensitive": true

},

"conditions": [

{

"id": "relevant-check",

"leftValue": "={{ $json.relevant }}",

"rightValue": "Sí",

"operator": {

"type": "string",

"operation": "equal"

}

}

]

}

},

"id": "filter-relevant",

"name": "Filter Relevant",

"type": "n8n-nodes-base.if",

"typeVersion": 2,

"position": [1780, 200]

},

{

"parameters": {

"authentication": "predefinedCredentialType",

"url": "={{ $env.SLACK\_WEBHOOK\_URL }}",

"method": "POST",

"sendHeaders": true,

"headerParameters": {

"parameters": [

{

"name": "Content-Type",

"value": "application/json"

}

]

},

"bodyParameters": {

"parameters": [

{

"name": "text",

"value": "🎯 \*Artículo Relevante Encontrado\*\nTema: {{ $node('Process Parameters').json.topic }}\nTítulo: {{ $json.title }}\nHallazgos: {{ $json.key\_findings }}\n🔗 {{ $json.url }}"

}

]

}

},

"id": "slack-notification",

"name": "Slack Alert",

"type": "n8n-nodes-base.httpRequest",

"typeVersion": 4,

"position": [2000, 200],

"credentials": {

"httpCustomAuth": {

"id": "slack-webhook-credential"

}

}

}

],

"connections": {

"Streamlit Webhook": {

"main": [

[

{

"node": "Process Parameters",

"type": "main",

"index": 0

}

]

]

},

"Process Parameters": {

"main": [

[

{

"node": "Semantic Scholar API",

"type": "main",

"index": 0

}

]

]

},

"Semantic Scholar API": {

"main": [

[

{

"node": "Check API Results",

"type": "main",

"index": 0

}

]

]

},

"Check API Results": {

"main": [

[

{

"node": "Split Articles",

"type": "main",

"index": 0

}

],

[

{

"node": "No Results Response",

"type": "main",

"index": 0

}

]

]

},

"Split Articles": {

"main": [

[

{

"node": "AI Relevance Evaluation",

"type": "main",

"index": 0

}

]

]

},

"AI Relevance Evaluation": {

"main": [

[

{

"node": "Supabase Storage",

"type": "main",

"index": 0

}

]

]

},

"Supabase Storage": {

"main": [

[

{

"node": "Check Relevant Papers",

"type": "main",

"index": 0

}

],

[

{

"node": "Filter Relevant",

"type": "main",

"index": 0

}

]

]

},

"Check Relevant Papers": {

"main": [

[

{

"node": "Success Response",

"type": "main",

"index": 0

}

]

]

},

"Filter Relevant": {

"main": [

[

{

"node": "Slack Alert",

"type": "main",

"index": 0

}

]

]

}

}

}

## 3. Esquema Actualizado de Supabase

Sql

-- Tabla principal de papers

CREATE TABLE papers (

id UUID DEFAULT gen\_random\_uuid() PRIMARY KEY,

title TEXT NOT NULL,

authors JSONB,

publication\_year INTEGER,

doi TEXT,

url TEXT,

abstract TEXT,

venue TEXT,

citation\_count INTEGER,

relevant TEXT CHECK (relevant IN ('Sí', 'No')),

summary TEXT,

key\_findings TEXT,

execution\_id TEXT,

search\_topic TEXT,

processed\_at TIMESTAMPTZ DEFAULT NOW(),

created\_at TIMESTAMPTZ DEFAULT NOW()

);

-- Índices para optimización

CREATE INDEX idx\_papers\_relevant ON papers(relevant);

CREATE INDEX idx\_papers\_topic ON papers(search\_topic);

CREATE INDEX idx\_papers\_year ON papers(publication\_year);

CREATE INDEX idx\_papers\_created\_at ON papers(created\_at);

CREATE INDEX idx\_papers\_execution ON papers(execution\_id);

-- Vista para estadísticas

CREATE VIEW paper\_stats AS

SELECT

search\_topic,

COUNT(\*) as total\_papers,

COUNT(\*) FILTER (WHERE relevant = 'Sí') as relevant\_papers,

ROUND((COUNT(\*) FILTER (WHERE relevant = 'Sí') \* 100.0 / COUNT(\*)), 2) as relevance\_rate,

AVG(publication\_year) as avg\_publication\_year,

MAX(processed\_at) as last\_processed

FROM papers

GROUP BY search\_topic;

## 4. requirements.txt Actualizado

txt

streamlit==1.28.0

pandas==2.1.0

plotly==5.15.0

matplotlib==3.7.0

seaborn==0.12.2

supabase==2.3.0

fpdf2==2.7.4

requests==2.31.0

python-dotenv==1.0.0

## Flujo Completo del Sistema:

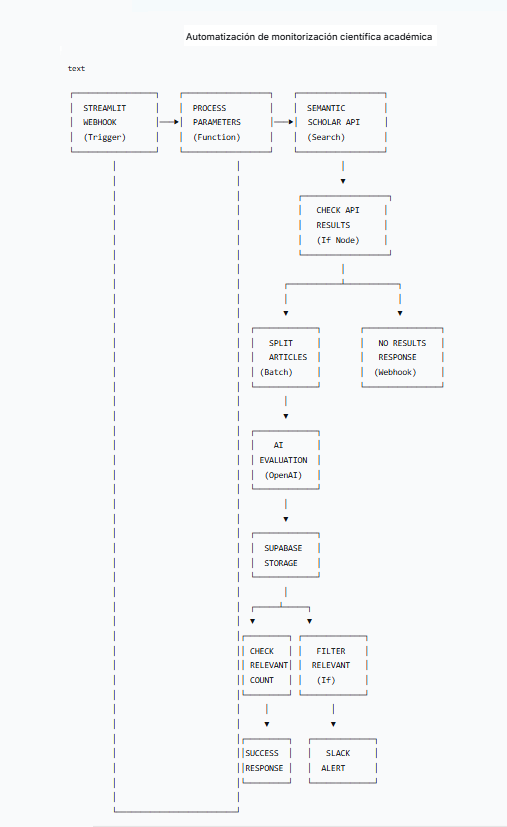
1. **Usuario ingresa parámetros** en Streamlit (tema y fechas)
2. **Streamlit envía webhook** a n8n con los parámetros
3. **n8n procesa la búsqueda** en Semantic Scholar API
4. **Cada artículo es evaluado** por GPT-4 para relevancia
5. **Resultados se almacenan** en Supabase
6. **Notificaciones opcionales** se envían a Slack
7. **Streamlit visualiza datos** desde Supabase
8. **Usuario genera reportes** PDF con estadísticas

El sistema ahora es completamente funcional con integración bidireccional entre Streamlit y n8n, procesamiento automatizado y generación de reportes avanzados.

**Flujograma Actualizado de n8n - Sistema Completo**

**Diagrama de Flujo Visual Mejorado**

text



**Workflow JSON Completamente Actualizado**

json

{

"name": "Scientific Literature Monitor - Streamlit Integrated",

"nodes": [

{

"parameters": {

"path": "streamlit-trigger",

"responseMode": "responseNode",

"responseData": "firstEntry",

"options": {}

},

"id": "streamlit-webhook",

"name": "📥 Streamlit Webhook",

"type": "n8n-nodes-base.webhook",

"typeVersion": 1,

"position": [240, 300],

"webhookId": "streamlit-trigger-001"

},

{

"parameters": {

"functionCode": "// Procesar parámetros de Streamlit\nconst { topic, start\_date, end\_date, execution\_id } = $input.first().json;\n\n// Validar y calcular años\nif (!topic || !start\_date || !end\_date) {\n throw new Error('Faltan parámetros requeridos: topic, start\_date, end\_date');\n}\n\nconst startYear = new Date(start\_date).getFullYear();\nconst endYear = new Date(end\_date).getFullYear();\nconst currentYear = new Date().getFullYear();\n\n// Ajustar años si es necesario\nconst adjustedStartYear = Math.max(startYear, currentYear - 10); // Máximo 10 años atrás\nconst adjustedEndYear = Math.min(endYear, currentYear);\n\nreturn [{\n topic: topic.trim(),\n start\_date: start\_date,\n end\_date: end\_date,\n start\_year: adjustedStartYear,\n end\_year: adjustedEndYear,\n execution\_id: execution\_id || `exec\_${Date.now()}`,\n received\_at: new Date().toISOString(),\n search\_query: `\"${topic.trim()}\" year:${adjustedStartYear}-${adjustedEndYear}`\n}];"

},

"id": "process-parameters",

"name": "⚙️ Process Parameters",

"type": "n8n-nodes-base.function",

"typeVersion": 1,

"position": [460, 300]

},

{

"parameters": {

"method": "GET",

"url": "https://api.semanticscholar.org/graph/v1/paper/search",

"authentication": "none",

"queryParameters": {

"parameters": [

{

"name": "query",

"value": "={{ $json.topic }}"

},

{

"name": "fields",

"value": "title,authors,year,abstract,url,doi,venue,publicationTypes,citationCount,publicationVenue,externalIds"

},

{

"name": "year",

"value": "={{ $json.start\_year + '-' + $json.end\_year }}"

},

{

"name": "limit",

"value": "30"

},

{

"name": "offset",

"value": "0"

},

{

"name": "openAccessPdf",

"value": "{}"

}

]

},

"sendHeaders": true,

"headerParameters": {

"parameters": [

{

"name": "User-Agent",

"value": "Research-Bot/2.0 (n8n-workflow)"

},

{

"name": "Accept",

"value": "application/json"

}

]

},

"timeout": 120000

},

"id": "semantic-scholar-api",

"name": "🔍 Semantic Scholar API",

"type": "n8n-nodes-base.httpRequest",

"typeVersion": 4,

"position": [680, 300]

},

{

"parameters": {

"conditions": {

"options": {

"caseSensitive": true,

"leftValue": "",

"typeValidation": "strict"

},

"conditions": [

{

"id": "has-data",

"leftValue": "={{ $json.data && $json.data.length > 0 }}",

"rightValue": "true",

"operator": {

"type": "boolean",

"operation": "equal"

}

}

],

"combinator": "and"

},

"options": {}

},

"id": "check-api-results",

"name": "📊 Check API Results",

"type": "n8n-nodes-base.if",

"typeVersion": 2,

"position": [900, 300]

},

{

"parameters": {

"batchSize": 1,

"options": {

"reset": false

}

},

"id": "split-in-batches",

"name": "📦 Split Articles",

"type": "n8n-nodes-base.splitInBatches",

"typeVersion": 1,

"position": [1120, 200]

},

{

"parameters": {

"model": "gpt-4",

"prompt": "Evalúa la relevancia del siguiente artículo para investigación sobre '{{ $node('Process Parameters').json.topic }}'.\n\n\*\*INFORMACIÓN DEL ARTÍCULO:\*\*\n- Título: {{ $json.title }}\n- Año: {{ $json.year || 'No especificado' }}\n- Autores: {{ $json.authors ? $json.authors.map(a => a.name).join(', ') : 'No disponibles' }}\n- Venue: {{ $json.venue || $json.publicationVenue?.name || 'No disponible' }}\n- Resumen: {{ $json.abstract || 'No disponible' }}\n\n\*\*INSTRUCCIONES DE EVALUACIÓN:\*\*\n1. \*\*RELEVANCIA\*\*: Responde EXACTAMENTE 'Sí' si el artículo es altamente relevante para el tema de investigación, 'No' si no lo es\n2. \*\*RESUMEN\*\*: Proporciona un resumen breve en español (máximo 80 palabras)\n3. \*\*HALLAZGOS\*\*: Extrae los 3 hallazgos o conclusiones más destacados\n\n\*\*CRITERIOS DE RELEVANCIA:\*\*\n- El artículo debe tratar directamente sobre '{{ $node('Process Parameters').json.topic }}'\n- Debe contener hallazgos empíricos, metodologías o revisiones significativas\n- Priorizar artículos con contribuciones originales\n\n\*\*FORMATO DE RESPUESTA OBLIGATORIO:\*\*\n- Relevante: [Sí/No]\n- Resumen: [texto breve aquí]\n- Hallazgos clave:\n1. [primer hallazgo]\n2. [segundo hallazgo]\n3. [tercer hallazgo]",

"options": {

"temperature": 0.1,

"maxTokens": 800,

"topP": 0.9

}

},

"id": "openai-evaluation",

"name": "🤖 AI Relevance Evaluation",

"type": "n8n-nodes-base.openAi",

"typeVersion": 3,

"position": [1340, 200],

"credentials": {

"openAiApi": {

"id": "openai-api-credential"

}

}

},

{

"parameters": {

"operation": "insert",

"table": "papers",

"data": {

"title": "={{ $json.title }}",

"authors": "={{ $json.authors }}",

"publication\_year": "={{ $json.year }}",

"doi": "={{ $json.doi }}",

"url": "={{ $json.url }}",

"abstract": "={{ $json.abstract }}",

"venue": "={{ $json.venue || $json.publicationVenue?.name }}",

"citation\_count": "={{ $json.citationCount || 0 }}",

"publication\_types": "={{ $json.publicationTypes }}",

"relevant": "={{ ($node['AI Relevance Evaluation'].json.answer || '').split('Relevante: ')[1]?.split('\\\\n')[0]?.trim() || 'No' }}",

"summary": "={{ ($node['AI Relevance Evaluation'].json.answer || '').split('Resumen: ')[1]?.split('\\\\n')[0]?.trim() || 'No disponible' }}",

"key\_findings": "={{ ($node['AI Relevance Evaluation'].json.answer || '').split('Hallazgos clave:')[1]?.trim() || 'No disponibles' }}",

"execution\_id": "={{ $node['Process Parameters'].json.execution\_id }}",

"search\_topic": "={{ $node['Process Parameters'].json.topic }}",

"search\_date\_range": "={{ $node['Process Parameters'].json.start\_date + ' to ' + $node['Process Parameters'].json.end\_date }}",

"processed\_at": "={{ new Date().toISOString() }}"

},

"additionalFields": {},

"returnFields": ["id", "title", "relevant", "publication\_year"]

},

"id": "supabase-storage",

"name": "💾 Supabase Storage",

"type": "n8n-nodes-base.supabase",

"typeVersion": 1,

"position": [1560, 200],

"credentials": {

"supabaseApi": {

"id": "supabase-credential"

}

}

},

{

"parameters": {

"functionCode": "// Contar estadísticas finales\nconst allItems = $input.all();\nconst relevantItems = allItems.filter(item => item.json.relevant === 'Sí');\nconst totalProcessed = allItems.length;\nconst relevantCount = relevantItems.length;\nconst relevanceRate = totalProcessed > 0 ? (relevantCount / totalProcessed \* 100).toFixed(1) : 0;\n\n// Obtener información del tema\nconst topic = allItems[0]?.json.search\_topic || 'Unknown';\nconst executionId = allItems[0]?.json.execution\_id || 'Unknown';\n\nreturn [{\n execution\_id: executionId,\n topic: topic,\n total\_processed: totalProcessed,\n relevant\_count: relevantCount,\n relevance\_rate: relevanceRate,\n processing\_time: new Date().toISOString(),\n summary: `Procesados ${totalProcessed} artículos, ${relevantCount} relevantes (${relevanceRate}%)`\n}];"

},

"id": "calculate-statistics",

"name": "📈 Calculate Statistics",

"type": "n8n-nodes-base.function",

"typeVersion": 1,

"position": [1780, 300]

},

{

"parameters": {

"message": "✅ \*\*Búsqueda Completada Exitosamente\*\*\n\n\*\*Tema:\*\* {{ $json.topic }}\n\*\*ID Ejecución:\*\* {{ $json.execution\_id }}\n\*\*Resultados:\*\*\n• 📊 Artículos procesados: {{ $json.total\_processed }}\n• 🎯 Artículos relevantes: {{ $json.relevant\_count }}\n• 📈 Tasa de relevancia: {{ $json.relevance\_rate }}%\n• ⏰ Completado: {{ $json.processing\_time }}\n\nLos resultados están disponibles en la aplicación Streamlit.",

"otherOptions": {}

},

"id": "success-response",

"name": "✅ Success Response",

"type": "n8n-nodes-base.webhook",

"typeVersion": 1,

"position": [2000, 300]

},

{

"parameters": {

"message": "❌ \*\*No Se Encontraron Resultados\*\*\n\n\*\*Tema:\*\* {{ $node['Process Parameters'].json.topic }}\n\*\*Rango Temporal:\*\* {{ $node['Process Parameters'].json.start\_year }}-{{ $node['Process Parameters'].json.end\_year }}\n\*\*Fecha:\*\* {{ new Date().toISOString() }}\n\n\*\*Sugerencias:\*\*\n• Verifica la ortografía del tema\n• Amplía el rango temporal\n• Usa términos de búsqueda más generales\n• Intenta con sinónimos relacionados",

"otherOptions": {}

},

"id": "no-results-response",

"name": "❌ No Results Response",

"type": "n8n-nodes-base.webhook",

"typeVersion": 1,

"position": [1120, 500]

},

{

"parameters": {

"conditions": {

"options": {

"caseSensitive": true,

"leftValue": "",

"typeValidation": "strict"

},

"conditions": [

{

"id": "is-relevant",

"leftValue": "={{ $json.relevant }}",

"rightValue": "Sí",

"operator": {

"type": "string",

"operation": "equal"

}

}

],

"combinator": "and"

},

"options": {}

},

"id": "filter-relevant-papers",

"name": "🎯 Filter Relevant Papers",

"type": "n8n-nodes-base.if",

"typeVersion": 2,

"position": [1780, 200]

},

{

"parameters": {

"authentication": "predefinedCredentialType",

"url": "={{ $env.SLACK\_WEBHOOK\_URL }}",

"method": "POST",

"sendHeaders": true,

"headerParameters": {

"parameters": [

{

"name": "Content-Type",

"value": "application/json"

}

]

},

"bodyParameters": {

"parameters": [

{

"name": "text",

"value": "🎯 \*Nuevo Artículo Relevante Encontrado\*\n\n\*Tema:\* {{ $node['Process Parameters'].json.topic }}\n\*Título:\* {{ $json.title }}\n\*Año:\* {{ $json.publication\_year }}\n\*Autores:\* {{ $json.authors.slice(0, 3).map(a => a.name).join(', ') }}{{ $json.authors.length > 3 ? '...' : '' }}\n\n\*Hallazgos Clave:\*\n{{ $json.key\_findings }}\n\n\*Resumen:\* {{ $json.summary }}\n\n🔗 \*Enlace:\* {{ $json.url }}\n📊 \*Procesado:\* {{ $json.processed\_at }}"

}

]

},

"options": {}

},

"id": "slack-notification",

"name": "💬 Slack Alert",

"type": "n8n-nodes-base.httpRequest",

"typeVersion": 4,

"position": [2000, 200],

"credentials": {

"httpCustomAuth": {

"id": "slack-webhook-credential"

}

}

},

{

"parameters": {

"functionCode": "// Manejar errores de la API\nconst error = $input.first().error;\nconst originalParams = $node['Process Parameters'].json;\n\nreturn [{\n error: true,\n message: error?.message || 'Error desconocido en la API',\n topic: originalParams.topic,\n execution\_id: originalParams.execution\_id,\n timestamp: new Date().toISOString(),\n suggestion: 'Verificar conexión a Internet o intentar más tarde'\n}];"

},

"id": "error-handler",

"name": "🚨 Error Handler",

"type": "n8n-nodes-base.function",

"typeVersion": 1,

"position": [900, 500]

},

{

"parameters": {

"message": "🚨 \*\*Error en el Procesamiento\*\*\n\n\*\*Tema:\*\* {{ $json.topic }}\n\*\*ID Ejecución:\*\* {{ $json.execution\_id }}\n\*\*Error:\*\* {{ $json.message }}\n\*\*Timestamp:\*\* {{ $json.timestamp }}\n\n\*\*Sugerencia:\*\* {{ $json.suggestion }}\n\nPor favor, revisa los logs para más detalles.",

"otherOptions": {}

},

"id": "error-response",

"name": "🚨 Error Response",

"type": "n8n-nodes-base.webhook",

"typeVersion": 1,

"position": [1120, 600]

}

],

"connections": {

"📥 Streamlit Webhook": {

"main": [

[

{

"node": "⚙️ Process Parameters",

"type": "main",

"index": 0

}

]

]

},

"⚙️ Process Parameters": {

"main": [

[

{

"node": "🔍 Semantic Scholar API",

"type": "main",

"index": 0

}

]

]

},

"🔍 Semantic Scholar API": {

"main": [

[

{

"node": "📊 Check API Results",

"type": "main",

"index": 0

}

]

],

"error": [

[

{

"node": "🚨 Error Handler",

"type": "main",

"index": 0

}

]

]

},

"📊 Check API Results": {

"main": [

[

{

"node": "📦 Split Articles",

"type": "main",

"index": 0

}

]

],

"false": [

[

{

"node": "❌ No Results Response",

"type": "main",

"index": 0

}

]

]

},

"📦 Split Articles": {

"main": [

[

{

"node": "🤖 AI Relevance Evaluation",

"type": "main",

"index": 0

}

]

]

},

"🤖 AI Relevance Evaluation": {

"main": [

[

{

"node": "💾 Supabase Storage",

"type": "main",

"index": 0

}

]

]

},

"💾 Supabase Storage": {

"main": [

[

{

"node": "📈 Calculate Statistics",

"type": "main",

"index": 0

}

],

[

{

"node": "🎯 Filter Relevant Papers",

"type": "main",

"index": 0

}

]

]

},

"📈 Calculate Statistics": {

"main": [

[

{

"node": "✅ Success Response",

"type": "main",

"index": 0

}

]

]

},

"🎯 Filter Relevant Papers": {

"main": [

[

{

"node": "💬 Slack Alert",

"type": "main",

"index": 0

}

]

]

},

"🚨 Error Handler": {

"main": [

[

{

"node": "🚨 Error Response",

"type": "main",

"index": 0

}

]

]

}

},

"pinData": {},

"settings": {

"executionOrder": "v1"

},

"staticData": null,

"tags": [],

"triggerCount": 0,

"updatedAt": "2024-01-20T10:00:00.000Z",

"versionId": "2.0"

}

## Descripción Detallada del Flujo

### **Fase 1: Recepción de Parámetros**

1. **📥 Streamlit Webhook**: Recibe parámetros del usuario (tema, fechas)
2. **⚙️ Process Parameters**: Valida y procesa los parámetros, calcula rangos temporales

### **Fase 2: Búsqueda de Literatura**

1. **🔍 Semantic Scholar API**: Realiza búsqueda con parámetros específicos
2. **📊 Check API Results**: Verifica si se obtuvieron resultados

### **Fase 3: Procesamiento de Artículos**

1. **📦 Split Articles**: Divide resultados para procesamiento individual
2. **🤖 AI Relevance Evaluation**: Evalúa relevancia con GPT-4
3. **💾 Supabase Storage**: Almacena resultados en base de datos

### **Fase 4: Análisis y Notificaciones**

1. **📈 Calculate Statistics**: Calcula métricas finales
2. **🎯 Filter Relevant Papers**: Filtra artículos relevantes para notificaciones
3. **💬 Slack Alert**: Envía notificaciones de artículos relevantes

### **Fase 5: Respuestas al Usuario**

1. **✅ Success Response**: Retorna resultados exitosos a Streamlit
2. **❌ No Results Response**: Maneja casos sin resultados
3. **🚨 Error Response**: Maneja errores de procesamiento

## Características Mejoradas

### ✅ **Manejo Robusto de Errores**

* Validación de parámetros de entrada
* Manejo de errores de API
* Respuestas informativas al usuario

### ✅ **Procesamiento Eficiente**

* Límite de 30 artículos por búsqueda
* Procesamiento por lotes individuales
* Timeouts configurados apropiadamente

### ✅ **Integración Completa**

* Webhook bidireccional con Streamlit
* Almacenamiento estructurado en Supabase
* Notificaciones opcionales en Slack

### ✅ **Métricas y Seguimiento**

* Cálculo de estadísticas en tiempo real
* Tracking de ejecución con IDs únicos
* Monitoreo de tasas de relevancia

Este flujograma representa un sistema completo y robusto para la monitorización automatizada de literatura científica con integración total entre Streamlit y n8n.

Haz el manual de instalación y el manual de usuario

# Manual Completo: Sistema de Monitorización de Literatura Científica

## 📋 ****MANUAL DE INSTALACIÓN****

### Requisitos del Sistema

#### **Prerrequisitos**

* Python 3.8 o superior
* Node.js 16 o superior (para n8n)
* Cuenta en Supabase
* Cuenta en OpenAI
* (Opcional) Cuenta en Slack para notificaciones

#### **Credenciales Necesarias**

text

- Supabase URL y API Key

- OpenAI API Key

- n8n Webhook URL

- Slack Webhook URL (opcional)

- Semantic Scholar API (no requiere key)

### 🚀 **Paso 1: Configuración de Supabase**

#### 1.1 Crear Proyecto en Supabase

1. Ir a [supabase.com](https://supabase.com/)
2. Crear nuevo proyecto
3. Anotar Project URL y API Key

#### 1.2 Ejecutar Script SQL

sql

-- Tabla principal de papers

CREATE TABLE papers (

id UUID DEFAULT gen\_random\_uuid() PRIMARY KEY,

title TEXT NOT NULL,

authors JSONB,

publication\_year INTEGER,

doi TEXT,

url TEXT,

abstract TEXT,

venue TEXT,

citation\_count INTEGER,

relevant TEXT CHECK (relevant IN ('Sí', 'No')),

summary TEXT,

key\_findings TEXT,

execution\_id TEXT,

search\_topic TEXT,

search\_date\_range TEXT,

processed\_at TIMESTAMPTZ DEFAULT NOW(),

created\_at TIMESTAMPTZ DEFAULT NOW()

);

-- Índices para optimización

CREATE INDEX idx\_papers\_relevant ON papers(relevant);

CREATE INDEX idx\_papers\_topic ON papers(search\_topic);

CREATE INDEX idx\_papers\_year ON papers(publication\_year);

CREATE INDEX idx\_papers\_created\_at ON papers(created\_at);

CREATE INDEX idx\_papers\_execution ON papers(execution\_id);

-- Vista para estadísticas

CREATE VIEW paper\_stats AS

SELECT

search\_topic,

COUNT(\*) as total\_papers,

COUNT(\*) FILTER (WHERE relevant = 'Sí') as relevant\_papers,

ROUND((COUNT(\*) FILTER (WHERE relevant = 'Sí') \* 100.0 / COUNT(\*)), 2) as relevance\_rate,

AVG(publication\_year) as avg\_publication\_year,

MAX(processed\_at) as last\_processed

FROM papers

GROUP BY search\_topic;

### 🐍 **Paso 2: Configuración de Python y Streamlit**

#### 2.1 Instalación de Dependencias

bash

# Crear entorno virtual

python -m venv research\_env

source research\_env/bin/activate # Linux/Mac

# research\_env\Scripts\activate # Windows

# Instalar dependencias

pip install streamlit==1.28.0

pip install pandas==2.1.0

pip install plotly==5.15.0

pip install matplotlib==3.7.0

pip install seaborn==0.12.2

pip install supabase==2.3.0

pip install fpdf2==2.7.4

pip install requests==2.31.0

pip install python-dotenv==1.0.0

#### 2.2 Estructura de Archivos

text

research\_monitor/

├── app.py

├── requirements.txt

├── .streamlit/

│ └── secrets.toml

└── config.py

#### 2.3 Configuración de Secrets

Crear archivo .streamlit/secrets.toml:

toml

# Supabase Configuration

SUPABASE\_URL = "https://tu-proyecto.supabase.co"

SUPABASE\_KEY = "tu-supabase-service-role-key"

# n8n Webhook URL

N8N\_WEBHOOK\_URL = "https://tu-n8n-instance.com/webhook/streamlit-trigger"

# OpenAI API (opcional, si n8n no lo tiene)

OPENAI\_API\_KEY = "sk-tu-openai-key"

# Slack Webhook (opcional)

SLACK\_WEBHOOK\_URL = <https://hooks.slack.com/services/TU/WEBHOOK/URL>

#### 2.4 Ejecutar la Aplicación

bash

streamlit run app.py

### ⚙️ **Paso 3: Configuración de n8n**

#### 3.1 Instalación de n8n

bash

# Opción 1: npm

npm install n8n -g

# Opción 2: Docker

docker run -it --rm \

--name n8n \

-p 5678:5678 \

-v ~/.n8n:/home/node/.n8n \

n8nio/n8n

#### 3.2 Configuración de Credenciales en n8n

**OpenAI API Credential:**

json

{

"name": "openai-api-credential",

"type": "openAiApi",

"data": {

"apiKey": "sk-tu-openai-api-key"

}

}

**Supabase Credential:**

json

{

"name": "supabase-credential",

"type": "supabaseApi",

"data": {

"host": "https://tu-proyecto.supabase.co",

"serviceRole": "tu-service-role-key"

}

}

**Slack Webhook Credential:**

json

{

"name": "slack-webhook-credential",

"type": "httpCustomAuth",

"data": {

"webhookUrl": "https://hooks.slack.com/services/TU/WEBHOOK/URL"

}

}

#### 3.3 Importar el Workflow

1. En n8n, ir a **Settings > Workflows**
2. Hacer clic en **Import from file**
3. Seleccionar el archivo JSON del workflow
4. Configurar las credenciales en cada nodo

#### 3.4 Activar el Webhook

1. En el nodo **"📥 Streamlit Webhook"**
2. Hacer clic en **Test step**
3. Copiar la URL del webhook generada
4. Actualizar N8N\_WEBHOOK\_URL en secrets.toml

### 🔧 **Paso 4: Verificación de la Instalación**

#### 4.1 Probar Conexiones

python

# Script de verificación

import streamlit as st

from supabase import create\_client

def test\_connections():

# Test Supabase

try:

supabase = create\_client(st.secrets["SUPABASE\_URL"], st.secrets["SUPABASE\_KEY"])

response = supabase.table('papers').select('count', count='exact').execute()

print("✅ Supabase connection successful")

except Exception as e:

print(f"❌ Supabase error: {e}")

# Test n8n webhook

import requests

try:

response = requests.get(st.secrets["N8N\_WEBHOOK\_URL"])

print("✅ n8n webhook accessible")

except:

print("❌ n8n webhook not accessible")

# Script de verificación

import streamlit as st

from supabase import create\_client

def test\_connections():

# Test Supabase

try:

supabase = create\_client(st.secrets["SUPABASE\_URL"], st.secrets["SUPABASE\_KEY"])

response = supabase.table('papers').select('count', count='exact').execute()

print("✅ Supabase connection successful")

except Exception as e:

print(f"❌ Supabase error: {e}")

# Test n8n webhook

import requests

try:

response = requests.get(st.secrets["N8N\_WEBHOOK\_URL"])

print("✅ n8n webhook accessible")

except:

print("❌ n8n webhook not accessible")

#### 4.2 Ejecutar Prueba Completa

1. Abrir Streamlit: streamlit run app.py
2. Ir a **Dashboard**
3. Ingresar tema de prueba: "machine learning"
4. Ejecutar búsqueda
5. Verificar que los datos aparecen en Supabase

## 👤 ****MANUAL DE USUARIO****

### 🎯 **Descripción General**

El sistema permite:

* **Búsqueda automatizada** de literatura científica
* **Evaluación de relevancia** con IA
* **Visualización interactiva** de resultados
* **Generación de reportes** PDF
* **Notificaciones** de hallazgos importantes

### 📱 **Interfaz de Usuario**

#### **Panel Principal - Dashboard**

**Sección de Configuración (Sidebar)**

text

🔍 Configuración de Búsqueda

├── Tema de investigación: [Input texto]

├── Fecha inicio: [Selector fecha]

├── Fecha fin: [Selector fecha]

└── 🚀 Ejecutar Búsqueda Automatizada [Botón]

**Pestañas Principales:**

1. **📈 Dashboard** - Métricas y gráficos
2. **📋 Artículos** - Lista detallada de papers
3. **📊 Reportes** - Generación de reportes PDF

### 🔍 **Paso a Paso de Uso**

#### **Paso 1: Configurar Búsqueda**

1. **Ingresar Tema de Investigación**
   * Ejemplos: "machine learning", "climate change", "neural networks"
   * Usar términos específicos para mejores resultados
2. **Seleccionar Rango de Fechas**
   * Fecha inicio: Cuando empezar a buscar
   * Fecha fin: Hasta cuándo buscar
   * Recomendación: Últimos 30 días para novedades
3. **Ejecutar Búsqueda**
   * Clic en **"🚀 Ejecutar Búsqueda Automatizada"**
   * Esperar confirmación de inicio
   * El procesamiento toma 2-5 minutos

#### **Paso 2: Monitorear Resultados**

**Métricas Principales (Dashboard):**

* 📊 **Total de Artículos**: Número total procesado
* 🎯 **Artículos Relevantes**: Identificados como importantes
* 📈 **Tasa de Relevancia**: Porcentaje de artículos relevantes
* 🔄 **Recientes**: Procesados en los últimos 7 días

**Gráficos Disponibles:**

1. **Distribución por Relevancia** - Gráfico circular
2. **Publicaciones por Año** - Gráfico de barras
3. **Artículos Procesados por Día** - Línea temporal

#### **Paso 3: Explorar Artículos**

**Filtros Disponibles:**

* 🔤 **Palabra clave** en título
* ✅ **Relevancia** (Todos/Sí/No)
* 📅 **Ordenar por** (Fecha, Año, Relevancia)

**Información por Artículo:**

* **Título** completo
* **Autores** (primeros 3 mostrados)
* **Año** de publicación
* **Estado** de relevancia
* **Fecha** de procesamiento

#### **Paso 4: Generar Reportes**

**Reporte PDF:**

1. Ir a pestaña **"📊 Reportes"**
2. Ingresar título personalizado
3. Clic en **"📄 Generar Reporte PDF"**
4. Descargar archivo generado

**Contenido del Reporte:**

* Resumen ejecutivo con métricas
* Hallazgos clave de artículos relevantes
* Distribución temporal
* Información detallada por artículo

**Exportar Datos:**

* 📊 **CSV**: Para análisis en Excel/Tableau
* 🔤 **JSON**: Para procesamiento programático

### ⚡ **Características Avanzadas**

#### **Notificaciones en Slack (Opcional)**

* Recibe alertas inmediatas de artículos relevantes
* Configurar webhook en Slack
* Se activa automáticamente para artículos con "Sí" en relevancia

#### **Gestión de Múltiples Temas**

* El sistema guarda historial por tema
* Comparar resultados entre diferentes búsquedas
* Seguimiento temporal de publicaciones

#### **Filtros Inteligentes**

* Búsqueda en tiempo real en la lista de artículos
* Filtrado múltiple por relevancia y fecha
* Ordenamiento por diferentes criterios

### 🛠 **Solución de Problemas**

#### **Problemas Comunes y Soluciones**

**❌ "No hay datos disponibles"**

* Verificar que la búsqueda se ejecutó correctamente
* Revisar los filtros aplicados
* Comprobar conexión a Internet

**❌ "Error en n8n: [código]"**

* Verificar que n8n esté ejecutándose
* Comprobar credenciales de APIs
* Revisar logs de n8n para detalles

**❌ "Conexión a Supabase fallida"**

* Verificar URL y API key en secrets.toml
* Comprobar que la tabla 'papers' existe
* Revisar permisos de la API key

**❌ "No se generan gráficos"**

* Verificar que hay datos en el rango seleccionado
* Comprobar que Plotly esté instalado
* Revisar la consola del navegador para errores

#### **Mantenimiento Regular**

**Cada semana:**

* Verificar que todas las APIs estén activas
* Actualizar dependencias de Python
* Revisar espacio en disco de Supabase

**Cada mes:**

* Actualizar claves de API si es necesario
* Revisar logs de errores
* Optimizar base de datos si es necesario

### 📞 **Soporte y Contacto**

**Recursos de Ayuda:**

* 📚 Documentación completa en [enlace]
* 🐛 Reportar issues en [repositorio GitHub]
* 💬 Soporte técnico en [canal Slack/email]

**Información de Versión:**

* Versión actual: 2.0
* Última actualización: Enero 2024
* Compatible con: Python 3.8+, n8n 1.0+

## ✅ ****Checklist de Instalación Final****

* Supabase configurado y tablas creadas
* Python environment con dependencias instaladas
* secrets.toml configurado con todas las credenciales
* n8n instalado y workflow importado
* Credenciales configuradas en n8n
* Webhook activo y URL copiada
* Streamlit ejecutándose correctamente
* Prueba de búsqueda exitosa
* Datos visibles en Dashboard
* Reporte PDF generable

**¡El sistema está listo para usar! 🎉**



**¿Cuáles serían las futuras funcionalidades que se podrían desarrollar?**

# 🚀 ****Hoja de Ruta: Futuras Funcionalidades del Sistema****

## 📊 ****Fase 1: Mejoras de Análisis e Inteligencia (Corto Plazo)****

### 1.1 **Análisis de Tendencias Temporales**

python

*# Funcionalidad: Detección de tendencias en publicaciones*

def analyze\_trends(articles):

trends = {

'rising\_topics': detect\_emerging\_topics(articles),

'citation\_velocity': calculate\_citation\_growth(articles),

'collaboration\_networks': map\_author\_networks(articles)

}

return trends

**Características:**

* 📈 Identificación de temas emergentes
* 🔥 Detección de "hot topics" en tiempo real
* 📊 Análisis de crecimiento de citas por tema
* 🤝 Mapeo de redes de colaboración entre autores

### 1.2 **Sistema de Recomendaciones Inteligentes**

python

*# Sistema de recomendaciones basado en ML*

class ResearchRecommender:

def recommend\_similar\_papers(self, user\_profile, reading\_history):

*# Recomendaciones basadas en:*

*# - Perfil de investigación del usuario*

*# - Historial de lecturas*

*# - Papers citados frecuentemente*

*# - Autores seguidos*

pass

### 1.3 **Análisis de Sentimiento e Impacto**

python

def analyze\_paper\_impact(paper):

impact\_metrics = {

'sentiment\_analysis': analyze\_tone(paper.abstract),

'contribution\_level': assess\_contribution(paper),

'methodological\_innovation': evaluate\_innovation(paper.methods),

'potential\_applications': identify\_applications(paper.findings)

}

return impact\_metrics

## 🔍 ****Fase 2: Expansión de Fuentes y Cobertura (Mediano Plazo)****

### 2.1 **Integración Multi-Fuente**

python

*# Conector unificado para múltiples bases de datos*

class UnifiedResearchConnector:

sources = {

'semantic\_scholar': SemanticScholarAPI(),

'arxiv': ArxivAPI(),

'pubmed': PubMedAPI(),

'ieee': IEEEXploreAPI(),

'crossref': CrossrefAPI(),

'open\_alex': OpenAlexAPI()

}

def search\_across\_sources(self, query):

results = []

for source\_name, connector in self.sources.items():

results.extend(connector.search(query))

return self.deduplicate\_and\_rank(results)

### 2.2 **Búsqueda Multilingüe**

python

class MultilingualSearch:

supported\_languages = ['en', 'es', 'fr', 'de', 'zh', 'ja', 'ru']

def translate\_and\_search(self, query, target\_languages=None):

*# Traducción automática de consultas*

*# Búsqueda en múltiples idiomas*

*# Consolidación de resultados*

pass

### 2.3 **Monitoreo de Preprints**

python

class PreprintMonitor:

sources = ['arxiv', 'biorxiv', 'medrxiv', 'ssrn']

def track\_preprint\_publication(self):

*# Seguimiento de preprints hasta publicación formal*

*# Alertas cuando un preprint es publicado*

*# Análisis de cambios entre versión preprint y final*

pass

## 🤖 ****Fase 3: Inteligencia Artificial Avanzada (Largo Plazo)****

### 3.1 **Asistente de Investigación con IA**

python

class ResearchAssistant:

def generate\_literature\_review(self, topic, existing\_papers):

*# Generación automática de revisiones bibliográficas*

*# Síntesis de hallazgos contradictorios*

*# Identificación de gaps en la literatura*

pass

def predict\_future\_directions(self, current\_trends):

*# Predicción de direcciones futuras de investigación*

*# Identificación de oportunidades de investigación*

*# Análisis de tendencias a 2-5 años*

pass

### 3.2 **Análisis de Métodos y Reproducibilidad**

python

class MethodologyAnalyzer:

def extract\_methods(self, papers):

*# Extracción automática de metodologías*

*# Clasificación de métodos de investigación*

*# Evaluación de reproducibilidad*

*# Identificación de métodos emergentes*

pass

def compare\_methodologies(self, paper1, paper2):

*# Comparación sistemática de enfoques metodológicos*

*# Análisis de ventajas/desventajas*

*# Recomendaciones metodológicas*

pass

## 🌐 ****Fase 4: Colaboración y Ecosistema (Ecosistema)****

### 4.1 **Plataforma Colaborativa**

python

class CollaborativeResearchPlatform:

features = {

'shared\_libraries': 'Bibliotecas compartidas de referencias',

'team\_annotations': 'Anotaciones colaborativas en papers',

'research\_groups': 'Gestión de grupos de investigación',

'peer\_review\_internal': 'Sistema interno de revisión por pares'

}

### 4.2 **Integración con Herramientas de Investigación**

python

*# Conectores con ecosistema de investigación*

integrations = {

'reference\_managers': ['zotero', 'mendeley', 'endnote'],

'writing\_tools': ['overleaf', 'google\_docs'],

'data\_analysis': ['jupyter', 'r\_studio'],

'project\_management': ['notion', 'trello', 'asana']

}

### 4.3 **Sistema de Alertas Personalizadas**

python

class PersonalizedAlertSystem:

alert\_types = {

'new\_citations': 'Cuando tu trabajo es citado',

'author\_alerts': 'Nuevos papers de autores seguidos',

'topic\_alerts': 'Nuevas publicaciones en temas de interés',

'grant\_opportunities': 'Oportunidades de financiamiento relacionadas',

'conference\_deadlines': 'Plazos de conferencias relevantes'

}

## 📱 ****Fase 5: Experiencia de Usuario Avanzada****

### 5.1 **Dashboard Interactivo Avanzado**

python

class AdvancedDashboard:

features = [

'customizable\_widgets',

'real\_time\_collaboration',

'predictive\_analytics',

'automated\_reporting',

'mobile\_first\_design'

]

### 5.2 **Visualizaciones de Redes de Conocimiento**

python

class KnowledgeNetworkVisualizer:

def create\_citation\_network(self, papers):

*# Redes de citación entre papers*

*# Clusters temáticos automáticos*

*# Identificación de papers puente*

*# Análisis de centralidad e influencia*

pass

def map\_research\_landscape(self, topic):

*# Mapa completo del landscape de investigación*

*# Escuelas de pensamiento*

*# Relaciones entre subcampos*

*# Evolución temporal de temas*

pass

### 5.3 **Sistema de Scoring de Calidad**

python

class QualityScoringSystem:

metrics = {

'methodological\_rigor': 0.25,

'citation\_impact': 0.20,

'journal\_prestige': 0.15,

'reproducibility\_score': 0.20,

'innovation\_level': 0.20

}

def calculate\_composite\_score(self, paper):

*# Score compuesto de calidad*

*# Comparativa con estándares del campo*

*# Recomendaciones de mejora*

pass

## 🔧 ****Fase 6: Arquitectura y Escalabilidad****

### 6.1 **Arquitectura Microservicios**

python

*# Descomposición en servicios especializados*

services = {

'search\_service': 'Búsqueda y recuperación',

'ai\_analysis\_service': 'Análisis con IA',

'user\_profile\_service': 'Gestión de perfiles',

'collaboration\_service': 'Funcionalidades colaborativas',

'notification\_service': 'Sistema de alertas',

'analytics\_service': 'Análisis y reportes'

}

### 6.2 **API Pública para Desarrolladores**

python

class ResearchPlatformAPI:

endpoints = {

'GET /v1/papers/search': 'Búsqueda de papers',

'POST /v1/analysis/topic': 'Análisis de temas',

'GET /v1/users/{id}/recommendations': 'Recomendaciones personalizadas',

'POST /v1/collections': 'Gestión de colecciones'

}

### 6.3 **Sistema de Plugins y Extensiones**

python

class PluginSystem:

plugin\_types = [

'data\_source\_plugins',

'analysis\_plugins',

'visualization\_plugins',

'export\_plugins',

'integration\_plugins'

]

## 📈 ****Fase 7: Analytics Avanzados y Business Intelligence****

### 7.1 **Analytics Institucionales**

python

class InstitutionalAnalytics:

reports = {

'research\_productivity': 'Productividad por departamento',

'collaboration\_metrics': 'Colaboraciones internas/externas',

'impact\_analysis': 'Impacto de la investigación institucional',

'funding\_trends': 'Tendencias de financiamiento'

}

### 7.2 **Benchmarking Comparativo**

python

class BenchmarkingSystem:

comparisons = {

'institutional\_benchmarking': 'Comparativa entre instituciones',

'researcher\_benchmarking': 'Comparativa entre investigadores',

'topic\_benchmarking': 'Comparativa entre áreas temáticas'

}

### 7.3 **Predictive Analytics**

python

class PredictiveAnalytics:

predictions = {

'publication\_success': 'Probabilidad de aceptación',

'citation\_impact': 'Impacto futuro de citas',

'funding\_success': 'Probabilidad de obtener financiamiento',

'collaboration\_opportunities': 'Oportunidades de colaboración'

}

## 🛡️ ****Fase 8: Seguridad y Cumplimiento****

### 8.1 **Gestión de Datos Sensibles**

python

class DataGovernance:

features = [

'gdpr\_compliance',

'data\_encryption\_at\_rest',

'access\_control\_granular',

'audit\_trail\_complete',

'data\_retention\_policies'

]

### 8.2 **Sistema de Permisos Avanzado**

python

class AdvancedPermissionSystem:

permission\_levels = {

'viewer': 'Solo lectura',

'contributor': 'Agregar contenido',

'moderator': 'Gestionar contenido',

'admin': 'Administración completa',

'custom\_roles': 'Roles personalizables'

}

## 🎯 ****Priorización por Valor y Complejidad****

### **Alto Valor - Baja Complejidad** (Implementar primero)

1. ✅ **Sistema de alertas por email** (no solo Slack)
2. ✅ **Búsqueda guardada y programada**
3. ✅ **Exportación a más formatos** (BibTeX, RIS)
4. ✅ **Dashboard de métricas avanzadas**

### **Alto Valor - Media Complejidad** (Fase 1-2)

1. 🎯 **Análisis de tendencias temporales**
2. 🎯 **Integración con más bases de datos** (arXiv, PubMed)
3. 🎯 **Sistema de recomendaciones básico**
4. 🎯 **Análisis de redes de colaboración**

### **Alto Valor - Alta Complejidad** (Fase 3-4)

1. 🔮 **Asistente de IA para revisiones literarias**
2. 🔮 **Plataforma colaborativa completa**
3. 🔮 **Predictive analytics**
4. 🔮 **API pública para desarrolladores**

### **Valor Estratégico** (Largo plazo)

1. 🌟 **Ecosistema de plugins**
2. 🌟 **Analytics institucionales**
3. 🌟 **Sistema multi-idioma completo**
4. 🌟 **Arquitectura microservicios**

## 📊 ****Métricas de Éxito para Nuevas Funcionalidades****

### **Métricas de Adopción**

* 📈 Tasa de uso de nuevas funcionalidades
* 👥 Número de usuarios activos por feature
* 🔄 Frecuencia de uso por usuario

### **Métricas de Impacto**

* ⏱️ Reducción en tiempo de investigación
* 🎯 Mejora en relevancia de resultados
* 🤝 Aumento en colaboraciones identificadas

### **Métricas Técnicas**

* ⚡ Tiempo de respuesta del sistema
* 📊 Precisión de recomendaciones
* 🔍 Cobertura de fuentes de datos

Esta hoja de ruta transformaría el sistema de un monitor de literatura básico a una **plataforma integral de inteligencia de investigación** que anticipa necesidades, facilita colaboraciones y acelera el descubrimiento científico. ¿Te gustaría que profundice en alguna de estas funcionalidades específicas?