Fuentes de Datos -BAC

YoCampo Prototipo

Obtención de información de la Biblioteca Agropecuaria Colombiana BAC para alimentar el modelo RGA – YoCampo

Autores: Victor Manuel Mondragon Maca – Laboratorio de Datos SNUIRA

Versión:1.0

Fecha: 19 de septiembre de 2024

Unidad de Planificación Rural Agropecuaria (UPRA)
Calle 28 Nº 13-22, Torre C, piso 3. Edif. Palma Real. Bogotá, Colombia.
+57(601) 552 9820, 245 7307
www.upra.gov.co

Resumen

Este documento presenta una solución de interoperabilidad que integra la Biblioteca Agropecuaria de Colombia (BAC) [1] con la plataforma SIEMBRA a través de la API de PRIMO y Azure Blob Storage. La finalidad es automatizar la consulta, descarga, y almacenamiento de documentos científicos del sector agropecuario para su posterior procesamiento y análisis en el proyecto YoCampo, un modelo basado en inteligencia artificial con tecnología GPT-4o. La API de PRIMO [1] permite realizar búsquedas precisas de documentos utilizando parámetros como autor, tema y fecha, mientras que Azure Blob Storage [2] se utiliza para el almacenamiento de los archivos recuperados. Además, se aplica un análisis a los archivos PDF para determinar si contienen texto legible o requieren procesamiento OCR [3]. Esta arquitectura de interoperabilidad asegura la disponibilidad y organización eficiente de los documentos, facilitando su uso en el modelo YoCampo, que utiliza técnicas de retrieval-augmented generation (RAG) [4] para mejorar las respuestas automatizadas a preguntas del sector agropecuario. El estudio concluye que esta integración proporciona una solución escalable para la gestión de grandes volúmenes de información técnica y científica, optimizando el acceso y la utilización de estos datos para la innovación agropecuaria

Tabla de contenido

Índice de tablas

Índice de figuras

Lista de siglas y abreviaturas

API - Application Programming Interface

BAC - Biblioteca Agropecuaria de Colombia

SNIA - Sistema Nacional de Innovación Agropecuaria

OCR - Optical Character Recognition

CTI - Ciencia, Tecnología e Innovación

Glosario

Azure Cognitive Search: Un servicio de búsqueda en la nube con capacidades de inteligencia artificial, utilizado para realizar búsquedas avanzadas y recuperación de información.

Cognitive Services: Conjunto de servicios de inteligencia artificial que incluye procesamiento de lenguaje natural, visión artificial y servicios de habla.

Virtual Machines (VM): Instancias de servidores virtuales en la nube, altamente configurables y escalables.

Azure App Service: Plataforma totalmente gestionada para construir, desplegar y escalar aplicaciones web.

Azure Storage: Servicio de almacenamiento en la nube altamente escalable y duradero.

Azure Cosmos DB: Base de datos NoSQL globalmente distribuida, diseñada para ofrecer escalabilidad y disponibilidad altas.

Blob Storage: Servicio de almacenamiento de objetos de Azure, optimizado para almacenar grandes cantidades de datos no estructurados.

Azure OpenAI: Servicio que proporciona acceso a modelos avanzados de inteligencia artificial de OpenAI, como GPT-4, a través de la infraestructura de Azure.

YoCampo: Proyecto que utiliza múltiples servicios de Azure para ofrecer una solución robusta y escalable en el ámbito agropecuario.

Retrieval-Augmented Generation (RAG): Técnica que combina la recuperación de documentos relevantes con la generación de texto, mejorando la precisión y relevancia de las respuestas generadas por modelos de lenguaje.

API (Application Programming Interface): Conjunto de definiciones y protocolos que permiten la comunicación entre diferentes aplicaciones de software. En este contexto, la API de PRIMO permite realizar consultas y obtener datos desde el repositorio de la BAC.

Azure Blob Storage: Servicio de almacenamiento en la nube de Microsoft Azure que permite guardar grandes cantidades de datos no estructurados, como documentos, imágenes y archivos multimedia.

SIEMBRA: Plataforma del Sistema Nacional de Innovación Agropecuaria (SNIA) que facilita la gestión del conocimiento y el acceso a contenidos científicos y técnicos relacionados con el sector agropecuario.

PRIMO: Herramienta de descubrimiento bibliográfico que permite a los usuarios acceder a repositorios digitales y buscar documentos mediante consultas a través de una API.

MARC (Machine-Readable Cataloging): Formato estandarizado para representar información bibliográfica, utilizado en catálogos bibliográficos para describir recursos como libros, artículos y documentos.

Dublin Core: Conjunto de estándares de metadatos utilizado para describir recursos digitales, facilitando la interoperabilidad y la búsqueda eficiente de información en sistemas como PRIMO.

Introducción

Este documento tiene como finalidad proporcionar un análisis detallado de los costos asociados a la implementación de la solución YoCampo. Basada en la arquitectura RAG (Retrieval-Augmented Generation) y utilizando servicios avanzados de Azure, YoCampo es una aplicación que ofrece asistencia inteligente y personalizada a investigadores, técnicos, extensionistas y productores agropecuarios. Este documento presenta un proceso automatizado para consultar, descargar y almacenar archivos desde la Biblioteca Agropecuaria de Colombia (BAC) en Azure Blob Storage. Utilizando una API de BAC, se ejecutan búsquedas específicas.

Objetivos

- Implementar una solución automatizada para la consulta y descarga de archivos científicos desde la Biblioteca Agropecuaria de Colombia (BAC) mediante la integración de su API y el uso de consultas específicas, garantizando la recuperación eficiente de documentos relevantes.

- Configurar y utilizar Azure Blob Storage para el almacenamiento organizado de archivos descargados, facilitando su gestión a largo plazo en un entorno de almacenamiento en la nube escalable y accesible.

- Desarrollar un flujo de trabajo automatizado que integre servicios web y almacenamiento en la nube, permitiendo la interoperabilidad entre la plataforma de BAC y Azure, con el fin de optimizar la gestión de grandes volúmenes de información técnica y científica para el sector agropecuario

Alcance

El presente documento abarca la descripción y desarrollo de un proceso automatizado para la consulta, descarga y almacenamiento de documentos científicos desde la Biblioteca Agropecuaria de Colombia (BAC) hacia Azure Blob Storage. El alcance incluye:

- La configuración de la API de BAC para realizar búsquedas específicas, como en temáticas de interés agropecuario (por ejemplo, cacao).
- La implementación de un flujo de trabajo que permite la verificación de conectividad con el repositorio de AGROSAVIA y la posterior descarga de los archivos relevantes.
- La integración de Azure Blob Storage como plataforma de almacenamiento en la nube, con la configuración de contenedores y credenciales necesarias para la subida y gestión de archivos descargados.
- La documentación del proceso técnico, incluyendo la instalación de bibliotecas y scripts necesarios para la ejecución automática del flujo.
- La creación de una solución escalable y adaptable que facilita la gestión a largo plazo de grandes volúmenes de información científica y técnica relevante para el sector agropecuario.

1. Interoperabilidad BAC

El proyecto YoCampo, utiliza inteligencia artificial generativa para asistir a los actores del sector agropecuario con información actualizada y personalizada. La arquitectura RAG facilita la integración de estos componentes mediante la combinación de tecnologías de recuperación de información y generación de lenguaje natural. Al implementar una solución RAG, YoCampo puede ofrecer respuestas precisas y personalizadas a consultas específicas, de esta forma mejorando el acceso a la información para el beneficio de actores del Sistema Nacional de Innovación Agropecuaria (SNIA).

Este documento describe el proceso de automatización diseñado para consultar y recuperar archivos desde la interfaz API de la Biblioteca Agropecuaria de Colombia (BAC)[1] y almacenarlos de manera eficiente en una cuenta de Azure Blob Storage. Este proceso ha sido desarrollado con el objetivo de facilitar el acceso a información científica relevante, como publicaciones y documentos técnicos, desde el repositorio de AGROSAVIA.

A través de una secuencia de pasos automatizados, se verifica la conectividad con el repositorio, se realizan consultas específicas mediante la API de BAC (por ejemplo, sobre temas como el cacao), y se descargan los archivos pertinentes. Estos archivos se almacenan en contenedores de Azure Blob Storage para asegurar su disponibilidad y organización a largo plazo.

Este enfoque permite una integración fluida entre los servicios de almacenamiento en la nube de Azure y los datos científicos de BAC, proporcionando una solución escalable y automatizada para la gestión y almacenamiento de documentos

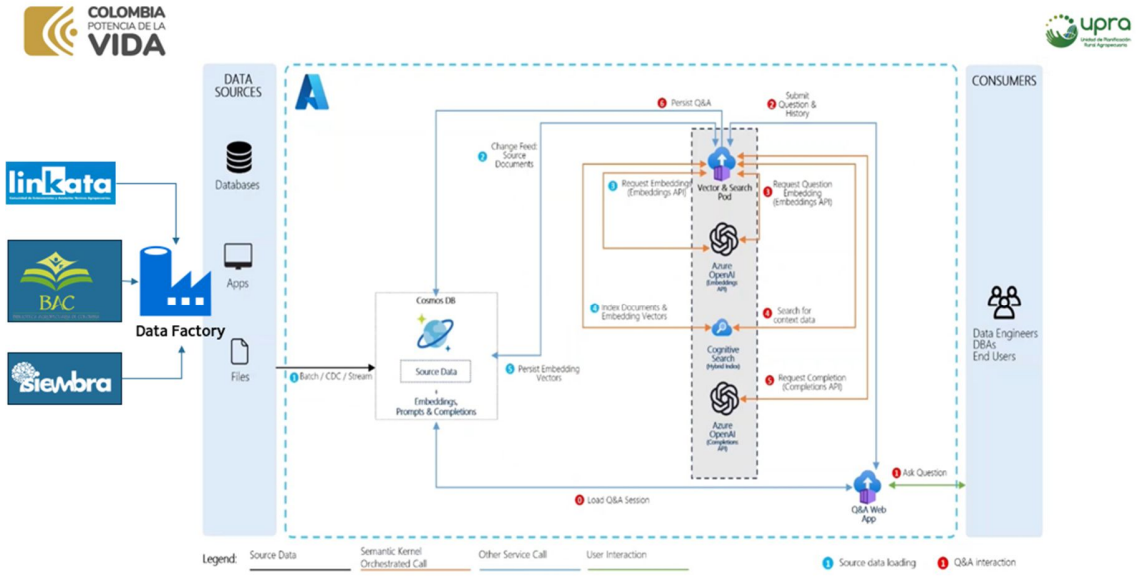
[1] https://www.agrosavia.co/biblioteca

A continuación, se describe la arquitectura de solución en la inter-operabilidad implementada en el prototipo experimental YoCampo.

1.1. Arquitectura de Solución

La arquitectura de solución*:* [5] representada en la Figura 1 para la interoperabilidad se compone de los siguientes elementos principales:

Figura 1. Arquitectura YoCampo – inter-operabilidad.



Fuente: [5](2024).

La arquitectura [6] mostrada representa el flujo de interoperabilidad entre diversas fuentes de datos y consumidores finales, permitiendo la integración de información proveniente de la Biblioteca Agropecuaria de Colombia (BAC) y otras fuentes, como Linkata y Siembra, para alimentar un sistema de generación de respuestas basado en inteligencia artificial, como GPT-4o (YoCampo) [3].

1.2. Descripción del Flujo de Interoperabilidad

1.2.1. Fuentes de Datos (Data Sources)

- Los datos provienen de bases de datos, aplicaciones y archivos almacenados en sistemas como Linkata, BAC, y Siembra. Estos datos incluyen documentos, metadatos y otras formas de conocimiento que son necesarias para alimentar el sistema.
- A través de Azure Data Factory, se realiza la ingesta de datos desde estas fuentes hacia el sistema central.

1.2.2. Procesamiento de Datos

- Los datos ingresados pasan a una base de datos en Cosmos DB que actúa como el almacén de datos estructurados. Aquí se almacenan los documentos y sus correspondientes vectores de embeddings (representaciones numéricas que facilitan la búsqueda semántica).
- En este paso se aplica el procesamiento de batch, CDC (Change Data Capture), o streaming, dependiendo de la naturaleza de los datos que están siendo sincronizados.
- Los datos son transformados en vectores y se preparan para ser utilizados en procesos de búsqueda semántica.

1.2.3. Indexación de Documentos y Vectores de Embedding

- Los documentos y sus vectores de embedding son indexados para que puedan ser consultados de manera eficiente mediante búsquedas semánticas.
- Este paso es fundamental para que el sistema pueda recuperar los documentos adecuados en base a las consultas del usuario.

1.2.4. Generación de Respuestas Basadas en IA

- El sistema utiliza dos APIs principales: Embeddings API y Completions API (Azure OpenAI), que permiten la búsqueda y recuperación de información relevante y la generación de respuestas basadas en el contenido de los documentos almacenados.
- Cuando un usuario realiza una pregunta, el sistema busca dentro de los vectores de embedding en un Vector & Search Pod, que utiliza la semántica para localizar el contenido más relevante.
- Posteriormente, la Completions API genera una respuesta a partir de los datos encontrados.

1.2.5. Interacción con los Consumidores

- Los usuarios, ya sea ingenieros de datos, DBAs, o usuarios finales, interactúan con el sistema a través de una Web App Q&A [2], donde realizan preguntas y reciben respuestas automatizadas.
- El flujo asegura que tanto la pregunta como el historial de consultas sean enviados al sistema para continuar optimizando las respuestas.

El análisis de los archivos PDF, descargados previamente desde la API de BAC y almacenados en Azure Blob Storage, es fundamental para que el sistema pueda procesar estos documentos. Los archivos son examinados para identificar si contienen texto (o si requieren OCR), y una vez procesados, son indexados en el sistema. Los vectores de embedding que representan el contenido semántico de estos documentos son cruciales para que el modelo GPT-4o (YoCampo) pueda generar respuestas informadas y contextualizadas basadas en consultas de los usuarios.

2. Flujo de Inter-operabilidad

En este apartado se describirá los flujos para la automatización en la obtención de los archivos fuentes de conocimiento que soportan el RGA – YoCampo.

2.1. Ingesta de Datos desde Fuentes Externas (BAC)

La ingesta de datos desde diversas fuentes externas es un componente fundamental en la arquitectura de interoperabilidad del proyecto YoCampo, ya que permite centralizar información valiosa proveniente de distintas plataformas y repositorios. En este apartado se detalla el proceso de integración y flujo de datos de tres fuentes clave: Biblioteca Agropecuaria de Colombia (BAC), Linkata, y Siembra.

La ingesta se realiza desde BAC Biblioteca Agropecuaria de Colombia (BAC). Siendo una de las principales fuentes de información utilizada en el sistema prototipo YoCampo, ya que alberga un amplio repositorio de documentos científicos y técnicos del sector agropecuario. El proceso de ingesta de datos desde BAC sigue los siguientes pasos:

- Acceso a la API de BAC [1]: A través de la configuración de la API de PRIMO, se permite la consulta de documentos directamente desde el repositorio de BAC. Los usuarios pueden realizar búsquedas utilizando parámetros específicos, como títulos, autores, temas o años de publicación.
- Extracción de Documentos: Una vez configurada la API, se extraen los documentos que cumplen con los criterios de búsqueda (por ejemplo, investigaciones sobre el cacao). Esta información incluye tanto metadatos (títulos, descripciones, autores) como los archivos completos en formato PDF.

- Almacenamiento Temporal: Los documentos extraídos se almacenan temporalmente en un repositorio local o en un servicio de nube (Azure Blob Storage), desde donde se procesarán antes de ser indexados en el sistema final.

- Transformación de Datos: Los archivos PDF se analizan para determinar si contienen texto legible o si requieren procesamiento adicional (como OCR) para convertir imágenes escaneadas en texto.

2.2. Consolidación de Datos en Azure

Una vez que los datos de BAC, Linkata y Siembra han sido extraídos, transformados y normalizados, estos se consolidan en **Azure Cosmos DB**. Este repositorio actúa como el centro de almacenamiento de los datos procesados y permite su posterior indexación y generación de *embeddings*, que serán utilizados por los algoritmos de inteligencia artificial para responder preguntas de los usuarios.

La **Data Factory de Azure** juega un papel crucial en este flujo, facilitando la orquestación de procesos y la integración de datos desde múltiples fuentes en tiempo real, con la posibilidad de manejar grandes volúmenes de datos de forma eficiente y escalable.

2.3. Configuración de la API de PRIMO para Consultas en BAC

La **API de PRIMO** [1] es el punto clave para la interoperabilidad entre SIEMBRA y BAC. PRIMO es una herramienta de descubrimiento que permite la consulta de repositorios digitales de BAC y de otras instituciones conectadas al sistema.

- **Acceso a la API**: Para conectar SIEMBRA con la API de PRIMO, es necesario configurar una serie de parámetros de acceso, incluyendo las claves de API, la URL del repositorio, y las credenciales de autenticación.
- **Consultas personalizadas**: La API permite realizar búsquedas mediante solicitudes HTTP, enviando parámetros de consulta específicos para recuperar documentos relacionados con temas agropecuarios.
- **Integración de Web Services**: Se configuran web services para que SIEMBRA pueda ejecutar consultas directamente desde su interfaz y recuperar los resultados obtenidos por PRIMO. Esto permite a los usuarios de SIEMBRA acceder a los repositorios de BAC sin necesidad de interactuar con la interfaz nativa de PRIMO.

2.4. Parámetros de Búsqueda en BAC

Los parámetros de búsqueda son esenciales para filtrar la información relevante en la API de PRIMO y garantizar que los usuarios accedan a los documentos más relevantes. BAC organiza sus documentos utilizando estándares bibliográficos como **MARC** y **Dublin Core**, que permiten categorizar la información de manera precisa.

- **Etiquetas MARC y Dublin Core**: Los parámetros de búsqueda se configuran utilizando etiquetas estándar como:
    - **dc.subject**: Para realizar búsquedas por tema o palabra clave.
    - **dc.title**: Para buscar por título de los documentos.
    - **dc.contributor**: Para buscar por autores o contribuyentes.
    - **dc.date**: Para limitar la búsqueda a documentos de un rango de fechas específico.
- **Ejemplos de consultas**:

- Búsqueda de documentos relacionados con el "cacao" dentro de la BAC: dc.subject=cacao.
- Búsqueda por autor en específico: dc.contributor.author=John Doe.

Estos parámetros permiten realizar búsquedas muy específicas y precisas, lo que resulta en una mayor relevancia de los resultados obtenidos.

2.5. Filtros y Tipologías de Documentos

Una de las funcionalidades más poderosas de la API de PRIMO es la capacidad de aplicar filtros y seleccionar tipologías de documentos para ajustar la búsqueda a las necesidades específicas del usuario.

- **Filtros disponibles**: La API de PRIMO [1] permite aplicar filtros como:
  - **Fecha de publicación**: Permite seleccionar documentos publicados en un rango de fechas específico.
  - **Departamento o región**: Filtro geográfico para buscar documentos relacionados con una región o departamento en particular.
  - **Tipo de documento**: Este filtro es útil para buscar artículos científicos, tesis, estudios técnicos, manuales, entre otros.
- **Tipologías de documentos**: Los documentos en BAC se clasifican en diversas categorías o tipologías como:
  - Libros
  - Artículos científicos
  - Tesis y trabajos de grado
  - Normas y reglamentos
  - Manuales y guías técnicas
  - Informes y estudios de mercado

Cada tipo de documento tiene su propia estructura de metadatos y formato, lo que facilita su búsqueda y recuperación a través de los filtros adecuados.

2.6. Limitaciones en la Recuperación de Datos

A pesar de la flexibilidad y precisión que ofrece la API de PRIMO, existen ciertas limitaciones en el proceso de recuperación de datos que deben ser consideradas al configurar las consultas en SIEMBRA.

Límite de resultados: La API de PRIMO [1] está diseñada para devolver un número limitado de resultados por consulta (normalmente 200 registros). Si la búsqueda devuelve más resultados, el sistema no los mostrará más allá de ese límite, lo que puede limitar el acceso a documentos importantes. Rendimiento de las consultas: Dependiendo de la cantidad de parámetros y la especificidad de las consultas, las respuestas de la API pueden ser más lentas, especialmente cuando se aplican múltiples filtros simultáneamente o cuando se solicitan grandes volúmenes de datos.

Acceso a documentos completos: En algunos casos, no todos los documentos disponibles en BAC están accesibles en texto completo. Algunos pueden estar restringidos a referencias o metadatos, lo que puede limitar la capacidad de análisis y procesamiento posterior para el modelo YoCampo.

Calidad de los metadatos: La calidad y consistencia de los metadatos en los documentos puede variar. Esto afecta la capacidad de realizar consultas precisas y puede requerir un procesamiento adicional de los datos para mejorar la consistencia y facilitar su uso en modelos de lenguaje como GPT-4o [5].

3. Implementación API BAC – Azure Storage

El archivo obtenido de la implementación OAI_FINAL_Storage_Account.ipynb contiene la implementación de la API PRIMO sobre los recursos del repositorio de la biblioteca BAC para ser copiados los archivos a la cuenta Storage Azure, que será la fuente de datos del modelo RGA YoCampo [4].

Este código está diseñado para obtener archivos desde un repositorio en línea, descargarlos y almacenarlos en una cuenta de almacenamiento en Azure Blob Storage, con el objetivo de usarlos en el entrenamiento de un modelo de lenguaje (GPT-4o) enfocado en la agricultura y el desarrollo rural. A continuación, te proporcionaré un análisis detallado de cada sección del código:

3.1. Instalación de bibliotecas

Tabla 1 Código instalación de librerías

python
!pip install azure-storage-blob !pip install requests !pip install beautifulsoup4 requests

Se instalan las bibliotecas necesarias:

- azure-storage-blob: Para interactuar con Azure Blob Storage.
- requests: Para realizar solicitudes HTTP.
- beautifulsoup4: Para analizar el contenido HTML.

3.2. Comprobación de conectividad

Tabla 2 Código para conectividad

python
import requests

```
def check_connectivity(url):
  try:
    response = requests.get(url, timeout=5)
    response.raise_for_status()
    print(f"{url} is reachable!")
  except requests.exceptions.RequestException as e:
    print(f"{url} is not reachable. Error: {e}")

check_connectivity('http://repository.agrosavia.co')
```

Esta función verifica si la URL del repositorio es accesible mediante una solicitud HTTP (requests.get). Si no lo es, se muestra el error correspondiente.

3.3. Importación de módulos y configuración

python
from bs4 import BeautifulSoup from azure.storage.blob import BlobServiceClient, BlobClient, ContainerClient import os

Aquí se importan los módulos necesarios. BeautifulSoup se usará para analizar HTML, y las clases de azure.storage.blob permiten interactuar con Azure Blob Storage para subir los archivos.

3.4. Configuración de la API y Azure Blob Storage

API_KEY: Clave de API para acceder al servicio de búsqueda.
BASE_URL: URL base de la API de búsqueda.
QUERY: Términos de búsqueda (en este caso, 'Cacao Plátano Cafe').
REPOSITORY_BASE_URL: URL base del repositorio.

```
API_KEY = 'l8xx066955af8e434edb84fc0fe545a1c1c6'
BASE_URL = 'https://api-na.hosted.exlibrisgroup.com/primo/v1/search'
QUERY = 'Cacao'
REPOSITORY_BASE_URL = 'https://repository.agrosavia.co'

blob_service_client = BlobServiceClient(account_url="https://mlailabsnuirastorage.blob.core.windows.net", credential="sp=racwd...")
container_name = "fileagrosavia"
container_client = blob_service_client.get_container_client(container_name)
```

Se configura la API de un sistema de búsqueda (BASE_URL), definiendo la palabra clave QUERY como "Cacao". Se usa BlobServiceClient para conectarse a Azure Blob Storage, utilizando una URL y una credencial de acceso (SAS Token). Luego, se define un contenedor donde se almacenarán los archivos descargados.

Se normaliza la consulta eliminando caracteres no alfanuméricos y convirtiéndola a minúsculas para crear un nombre de contenedor válido en Azure Blob Storage.

Tabla 3 Código búsqueda sobre la API

```
import re
from datetime import datetime

QUERY = 'Cacao Plátano Cafe'
normalized_query = re.sub(r'[^a-zA-Z0-9]', '', QUERY.lower())
from azure.storage.blob import BlobServiceClient

account_url = "https://storageyocampo.blob.core.windows.net"
```

```
credential = "tu_credencial_SAS"

blob_service_client = BlobServiceClient(account_url=account_url, credential=credential)
container_client = blob_service_client.create_container(container_name)
```

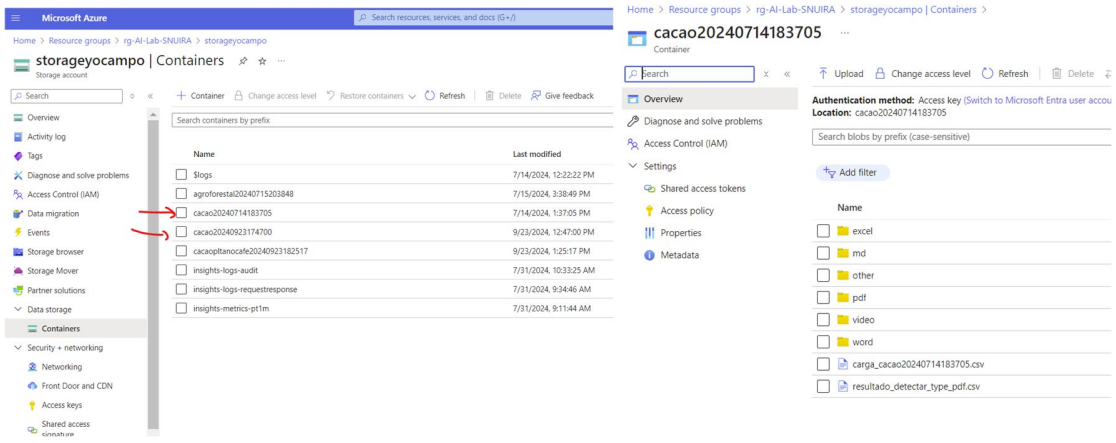
3.5. Creación de estructura de carpetas

Se crean carpetas dentro del contenedor para organizar los archivos según su tipo.

Tabla 4 Creación de destinos archivos

```
python
folder_structure = ["pdf", "excel", "word", "video", "other"]
for folder in folder_structure:
    blob_client = container_client.get_blob_client(f"{folder}/")
    blob_client.upload_blob('', overwrite=True)  # Crear carpeta vacía
```

Figura 2 Creación del Container con estructura de archivos



3.6. Exploración y descarga de archivos

Definición de funciones auxiliares

Extracción de metadatos: La función extract_metadata(addata) extrae metadatos de manera segura, manejando diferentes posibles claves en la respuesta de la API.

Tabla 5 exploración de la API

```
def extract_metadata(addata):
    return {
        'title': addata.get('btitle', [None])[0] if 'btitle' in addata else addata.get('title', [None])[0],
        'creator': addata.get('creator', [None])[0],
        'creationdate': addata.get('date', [None])[0] if 'date' in addata else addata.get('creationdate', [None])[0],
        'publisher': addata.get('publisher', [None])[0],
    }
```

Determinación del tipo de archivo: La función get_file_location_and_type(filename) clasifica los archivos según su extensión y determina su ubicación en el contenedor.

python

```
def get_file_location_and_type(filename):
    extension = filename.split('.')[-1].lower()
    if extension == 'pdf':
        return 'pdf', 'pdf'
    elif extension in ['doc', 'docx']:
        return 'word', 'word'
    elif extension in ['xls', 'xlsx']:
```

```
        return 'excel', 'excel'
    elif extension in ['mp4', 'avi', 'mov', 'flv', 'wmv']:
        return 'video', 'video'
    else:
        return 'other', 'other'
```

3.7. Búsqueda y descarga de archivos

Obtener el número total de elementos: Se realiza una solicitud inicial a la API para determinar cuántos resultados arroja la consulta.

```
params_total = {
    'vid': '57BAC_INST:BAC',
    'scope': 'RI_BAC',
    'q': f'any,contains,{QUERY}',
    'apikey': API_KEY,
    'limit': 1,
    'offset': 0,
    'sort': 'date'
}

response_total = requests.get(BASE_URL, params=params_total)
data_total = response_total.json()
total_items = data_total.get('info', {}).get('total', 0)

page_size = 50
pages = (total_items // page_size) + 1
```

Configurar la paginación: Se establece el tamaño de página y se calcula el número de páginas necesarias.

Iterar sobre las páginas de resultados: Para cada página, se realiza una solicitud a la API y se procesa cada documento.

```
metadata_list = []
uploaded_files = set()

for page in range(pages):
    params = {
        'vid': '57BAC_INST:BAC',
        'scope': 'RI_BAC',
        'q': f'any,contains,{QUERY}',
        'apikey': API_KEY,
        'limit': page_size,
        'offset': page * page_size,
        'sort': 'date'
    }

    response = requests.get(BASE_URL, params=params)
    data = response.json()

    for item in data.get('docs', []):
        # Procesamiento de cada documento
```

Descarga y subida de archivos: Se descarga cada archivo encontrado y se sube a Azure Blob Storage en la carpeta correspondiente, evitando duplicados.

Tabla 6 Descarga de archivos al Storage

```
if file_name not in uploaded_files:
    uploaded_files.add(file_name)

    # Descargar el archivo
    file_response = requests.get(file_url)
    if file_response.status_code == 200:
        # Determinar la ubicación y tipo de archivo
        location, file_type = get_file_location_and_type(file_name)
        blob_client = container_client.get_blob_client(f"{location}/{file_name}")
        blob_client.upload_blob(file_response.content, overwrite=True)
        print(f"Uploaded {file_name} to Azure Blob Storage in {location} with metadata: {metadata}")
    else:
        print(f"Failed to download {file_url}")

    # Actualizar metadatos con el nombre y tipo de archivo
    metadata.update({
        'idfile': pdf_name,
        'filename': file_name,
        'typefile': file_type
    })

    # Agregar metadatos a la lista
    metadata_list.append(metadata)
```

3.8. Guardado y subida de metadatos

Se guardan los metadatos recopilados en un archivo CSV y se sube al contenedor en Azure Blob Storage.

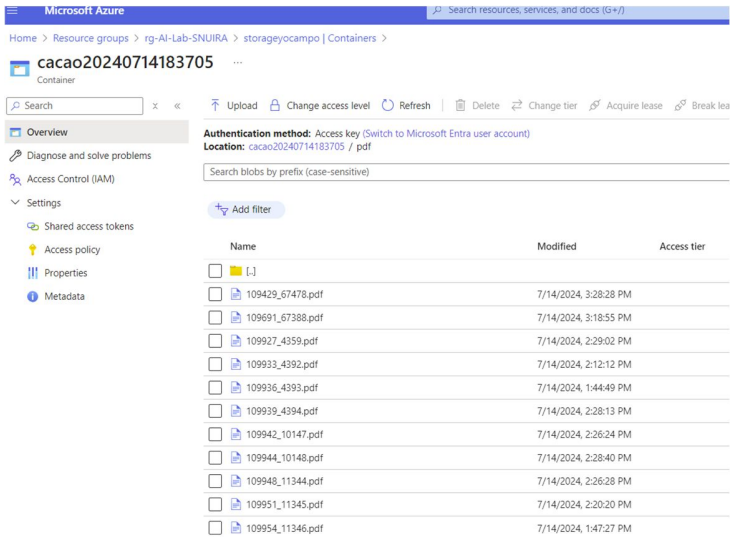
```
python
import pandas as pd

csv_filename = f"carga_{container_name}.csv"
metadata_df = pd.DataFrame(metadata_list)
metadata_df.to_csv(csv_filename, index=False)

# Subir el archivo CSV al contenedor
blob_client = container_client.get_blob_client(f"{csv_filename}")
with open(csv_filename, "rb") as data:
    blob_client.upload_blob(data, overwrite=True)

print(f"Metadata saved to {csv_filename} and uploaded to Azure Blob Storage")
```

Figura 3 Archivos PDF obtenidos



3.9. Identificación de videos

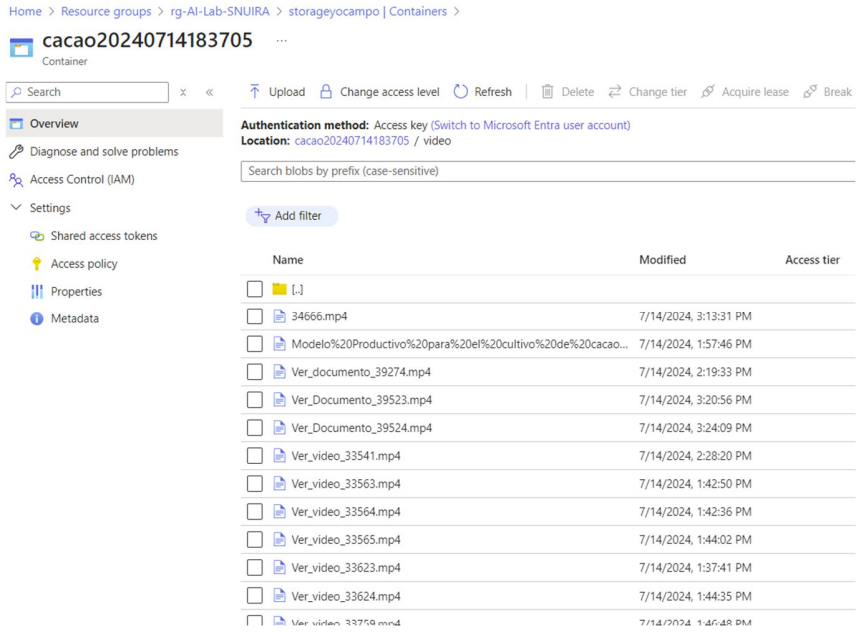
Se listan los archivos almacenados en la carpeta de videos dentro del contenedor para verificar su correcto almacenamiento.

Tabla 7 Validación de archivos de video

python

from azure.storage.blob import BlobServiceClient
 	
# Configuración de Azure Blob Storage 	
account_url = "https://storageyocampo.blob.core.windows.net"
credential = "tu_credencial_SAS"
folder_name = "video/"
 	
# Crear el cliente de servicio de blob 	
blob_service_client = BlobServiceClient(account_url=account_url, credential=credential)
container_client = blob_service_client.get_container_client(container_name)
 	
# Listar archivos en la carpeta de video 	
print(f"Archivos en la ruta {container_name}/{folder_name}:")
blob_list = container_client.list_blobs(name_starts_with=folder_name)
for blob in blob_list:
  print(blob.name)

Figura 4 Archivos de video identificados



En resumen, el archivo Nootebook ejecuta:

Instalación y configuración: Se instalan las librerías necesarias y se configuran las variables para conectarse a la API y a Azure Blob Storage.

Búsqueda en la API: Se realiza una búsqueda de documentos relacionados con los términos 'Cacao', 'Plátano' y 'Cafe' utilizando la API proporcionada.

Descarga de archivos: Se exploran los resultados y se descargan los archivos asociados desde el repositorio de Agrosavia.

Clasificación y almacenamiento: Los archivos se clasifican según su tipo (PDF, Word, Excel, Video, Otros) y se suben a Azure Blob Storage en las carpetas correspondientes.

Recopilación de metadatos: Se extraen metadatos relevantes de cada documento y se almacenan en un archivo CSV.

Verificación: Se verifica el contenido almacenado en Azure Blob Storage, especialmente en la carpeta de videos.

3.10. Validación de archivos PDF

El objetivo principal del notebook Tipo_PDF_Imagen_texto.ipynb es analizar y clasificar una colección de archivos PDF almacenados en un sistema de almacenamiento (como Azure Blob Storage) para determinar si el contenido de cada archivo es texto o una imagen. Esto es crucial para definir el procesamiento adecuado de los documentos dentro del flujo de trabajo del proyecto YoCampo, donde los documentos de texto pueden ser directamente indexados y utilizados por los modelos de inteligencia artificial (IA), mientras que los archivos que contienen imágenes necesitan un procesamiento adicional mediante OCR (Reconocimiento Óptico de Caracteres).

La clasificación de los archivos PDF en función de si contienen texto o imágenes facilita la preparación de datos para el entrenamiento de modelos de lenguaje como GPT-4o, asegurando que los datos se procesen adecuadamente antes de ser utilizados por los modelos de YoCampo

Seguidme se procede a describir el código fuente creado para el Script de identificación de archivos PDF

Carga de Bibliotecas:

La biblioteca `pandas` se importa para gestionar y manipular los datos resultantes del análisis de los archivos PDF. Se utiliza para cargar y procesar el archivo CSV que contiene los resultados de la clasificación de los PDFs.

Python
`import pandas as pd`

2. Cargar el archivo CSV con los resultados:

En esta sección, se carga un archivo CSV que contiene la información procesada previamente. Este archivo almacena detalles sobre los PDFs, como su nombre, tamaño y tipo (texto o imagen). El archivo se carga en un DataFrame de `pandas`, que permite manipular fácilmente los datos.

```
Python
resultado_csv_path='/tmp/resultado_detectar_type_pdf.csv'
df_resultado = pd.read_csv(resultado_csv_path)
```

Contar la cantidad de PDFs de texto e imagen:

Aquí, se utiliza la función `value_counts()` de `pandas` para contar cuántos archivos PDF son de tipo texto y cuántos son de tipo imagen. Se hace sobre la columna 'tipo' del DataFrame, donde se almacena esta clasificación (1 para texto y 0 para imagen).

```
Python
conteo_tipos = df_resultado['tipo'].value_counts()
```

Obtener las cantidades de PDFs de texto e imagen:

En esta parte, se extraen los conteos de cada tipo de archivo. `get(1, 0)` busca la cantidad de PDFs clasificados como texto (valor 1) y devuelve 0 si no hay ninguno. Del mismo modo, `get(0, 0)` busca la cantidad de PDFs clasificados como imagen (valor 0).

```
Python
cantidad_texto=conteo_tipos.get(1,0)
cantidad_imagen = conteo_tipos.get(0, 0)
```

Mostrar los resultados: Finalmente, se imprimen los resultados en la consola. Se muestran las cantidades totales de archivos PDF que contienen texto y los que contienen imágenes, proporcionando una visión clara del balance entre ambos tipos en la colección de archivos.

```
Python
print(f"Cantidad de PDFs de texto:{cantidad_texto}")
print(f"Cantidad de PDFs de imagen: {cantidad_imagen}")
```

Tabla 8 Ejemplo de resultados Type PDF

nombre_archivo	peso_kbytes	Validación	Tipo de PDF
pdf/109429_67478.pdf	57.342.451.171.875	1	texto
pdf/109691_67388.pdf	60.106.435.546.875	1	texto
pdf/109927_4359.pdf	770.775.390.625	1	texto
pdf/109933_4392.pdf	11.264.892.578.125	1	texto
pdf/109936_4393.pdf	28.790.625	1	texto
pdf/109939_4394.pdf	187.498.046.875	1	texto
pdf/109942_10147.pdf	36.327.734.375	0	imagen
pdf/109944_10148.pdf	102.394.921.875	1	texto
pdf/109948_11344.pdf	2.869.736.328.125	1	texto
pdf/110626_55408.pdf	22.036.455.078.125	1	texto
pdf/110755_67942.pdf	1.512.208.984.375	1	texto
pdf/110791_67985.pdf	31.952.080.078.125	0	imagen
pdf/110819_55416.pdf	44.455.947.265.625	1	texto
pdf/110822_55417.pdf	11.577.919.921.875	1	texto
pdf/110825_55418.pdf	60.304.130.859.375	1	texto
pdf/110828_55419.pdf	6.143.646.484.375	1	texto
pdf/110831_55420.pdf	22.931.123.046.875	1	texto
pdf/110834_55421.pdf	22.718.349.609.375	1	texto

pdf/110846_67995.pdf	31.452.841.796.875	1	texto
pdf/13600.pdf	317.947.265.625	1	texto
pdf/17567.pdf	29.738.662.109.375	1	texto
pdf/17597.pdf	27.128.388.671.875	1	texto
pdf/20844_1.pdf	37.680.443.359.375	1	texto
pdf/20909_109.pdf	681.298.828.125	0	imagen
pdf/20922_124.pdf	13.281.640.625	0	imagen
pdf/21431_1354.pdf	2.534.095.703.125	0	imagen
pdf/21788_1906.pdf	1.450.630.859.375	0	imagen
pdf/21933_2256.pdf	10.410.384.765.625	0	imagen
pdf/22320_2985.pdf	4.308.974.609.375	1	texto
pdf/22453_3226.pdf	22.093.037.109.375	0	imagen
pdf/22498_3275.pdf	907.275.390.625	1	texto
pdf/22544_1.pdf	178.752.734.375	1	texto
pdf/22712_3668.pdf	28.671.552.734.375	0	imagen
pdf/22784_3775.pdf	6.976.142.578.125	0	imagen
pdf/22785_3776.pdf	3.536.609.375	0	imagen
pdf/22786_3777.pdf	5.359.599.609.375	0	imagen

Análisis Estadístico de los Archivos PDF:

Distribución de tipos de archivos PDF:

- Archivos de texto (tipo 1): 848

- Archivos de imagen (tipo 0): 113

Estadísticas básicas del tamaño de los archivos PDF (en KB):

- Tamaño promedio de los archivos: 4896.87 KB

- Tamaño mínimo de los archivos: 3.58 KB

- Tamaño máximo de los archivos: 411418.76 KB

- Desviación estándar del tamaño de los archivos: 15877.11 KB

- Mediana del tamaño de los archivos: 2440.01 KB

4. Descripción Detallada del Código

Este notebook automatiza la tarea de revisar una colección de archivos PDF para determinar si cada uno es de tipo texto o imagen. La identificación correcta es crucial para decidir el tipo de procesamiento adicional que se requiere: los archivos de texto pueden ser utilizados inmediatamente por los modelos de lenguaje del proyecto YoCampo, mientras que los archivos de imágenes deben pasar por un proceso de OCR para extraer el contenido legible. Esta automatización ahorra tiempo y asegura que los datos se preparen adecuadamente para su uso en sistemas de inteligencia artificial.

El código sigue un enfoque secuencial en el que primero se cargan las bibliotecas necesarias, luego se procesa el archivo CSV con los resultados de clasificación, y finalmente se cuentan y muestran las cantidades de archivos de texto e imagen. Esta estructura permite un flujo de trabajo eficiente y asegura que los documentos estén correctamente organizados para su posterior uso en el entrenamiento de modelos de IA.

5. Conclusiones

- Automatización [3] y Eficiencia en la Gestión de Datos, El código analizado implementa una solución automatizada para la obtención, procesamiento y almacenamiento de archivos desde la Biblioteca Agropecuaria de Colombia (BAC) hacia Azure Blob Storage. Este proceso facilita la ingesta eficiente de grandes volúmenes de documentos científicos, garantizando su gestión centralizada y el acceso rápido a datos estructurados en la nube, lo que es esencial para el proyecto YoCampo.
- Interoperabilidad y Flexibilidad de Consultas API La configuración de la API de PRIMO, integrada con la plataforma SIEMBRA y BAC, permite realizar consultas flexibles y precisas. Esto asegura que los usuarios puedan acceder a documentos relevantes de manera rápida y personalizada, ajustando los parámetros de búsqueda según temas, fechas, autores y más. Esta capacidad de interoperabilidad es clave para enriquecer el modelo LLM YoCampo con información actualizada y de alta calidad.
- Preparación de Datos para Modelos de IA,El análisis de los archivos PDF, incluyendo la verificación de si contienen texto o imágenes, es un paso crucial para garantizar que los datos sean utilizables en el entrenamiento de modelos de lenguaje como GPT-4o. La capacidad de identificar y procesar estos archivos asegura que el sistema pueda integrar documentos de BAC directamente en el modelo YoCampo, mejorando así la precisión y relevancia de las respuestas generadas a partir de consultas de los usuarios.

6. Referencias

[1] Ex Libris, «Primo RESTful API Documentation,» 2024. [En línea]. Available: https://developers.exlibrisgroup.com/primo/apis/.

[2] Microsoft-architecture, «Architecting applications on Azure,» 20 06 2023. [En línea]. Available: https://learn.microsoft.com/en-us/azure/architecture/.

[3] Mondragon, V.M y García-Díaz, V, «Adaptive contents for interactive TV guided by machine learning based on predictive sentiment analysis of data,» *Springer.*

[4] B. Peng y C. Li, «Instruction Tuning with GPT-4,» *arXiv,* 2023.

[5] Microsoft Azure AI, «github - Vector-Search-AI-Assistant,» github , Marzo 2024. [En línea]. Available: https://github.com/Azure/Vector-Search-AI-Assistant/tree/cognitive-search-vector. [Último acceso: mayo 2024].

[6] microsoft, «Arquitectura de análisis moderno con Azure Databricks,» 23 Sep 2023. [En línea]. Available: https://learn.microsoft.com/es-es/azure/architecture/solution-ideas/articles/azure-databricks-modern-analytics-architecture.

Anexos

Anexo 1. MODELO DE INTEROPERABILIDAD BAC-SIEMBRA (AGROSAVIA)

El sistema de información SIEMBRA facilita el relacionamiento entre los actores del sector agropecuario desde la generación de nuevos conocimientos hasta su vinculación.

SIEMBRA es la Plataforma de información para la gestión del conocimiento en Ciencia, Tecnología e Innovación, (CTI) del Sistema Nacional de Innovación Agropecuaria, (SNIA). Aquí podrás encontrar contenidos relacionados con las dinámicas de ciencia, tecnología e innovación del sector e información estadística y referencial para la toma de decisiones informadas.

Cuenta con diferentes parcelas dentro de las cuales se encuentran Biblioteca Agropecuaria de Colombia – BAC, indicadores de CTI, demandas, entre otros componentes, como una herramienta de soporte documental a los proyectos y ofertas tecnológicas del sector. Para esto, la BAC como fuente principal de almacenamiento de producción de la Corporación colombiana de investigación agropecuaria AGROSAVIA, catalogará y centralizará todos los tipos de documentos generados por la corporación, del mismo modo integrará los documentos generados y/o catalogados por otras instituciones del sector.

MODELO REQUERIDO

Partiendo de la premisa que el descubridor actual de la BAC, a través de la plataforma PRIMO, tiene acceso a los repositorios digitales de otras bibliotecas, y a las fuentes propias de la Biblioteca Agropecuaria de Colombia, se requiere la configuración de una herramienta o API que permita la interoperabilidad de Primo con Siembra.

Por lo tanto, para dicha configuración inicialmente mediante el web service se realizó una configuración con los siguientes parámetros de consulta tanto en MARC como en dublin core:

Parámetros de consulta	Etiquetas MARC	Dublincore	XOAI	Campo de búsqueda creado
	650	dc.subject.agrovoc[spa]	<element name="subject"> <element name="agrovoc"> <element name="spa"> <field name="value">	Código 'lsr01'
Producto (Obligatorio)	653	dc.subject.proposal[spa] dc.subject.proposal[eng]	<element name="subject"> <element name="proposal"> <element name="spa"> <field name="value"> <element name="eng"> <field name="value">	
		dc.description.productionsystems[spa]	<element name="description"> <element name="productionsystems"> <element name="spa"> <field name="value">	
	245	dc tittle[spa] dc tittle[eng]	<element name="title"> <element name="spa"> <field name="value"> <element name="eng"> <field name="value">	
Departamento (opcional)	651	dc.coverage.department[spa] dc.coverage.department	<element name="coverage"> <element name="department"> <element name="spa"> <field name="value">	Código 'lsr02'
	245	dc tittle[spa] dc tittle[eng]	<element name="title"> <element name="spa"> <field name="value"> <element name="eng"> <field name="value">	
Autor (opcional)	100 - 700	dc.contributor.author	<element name="creator"> <field name="value">	Código 'lsr03'

	110 - 710	dc.contributor.corporatename[spa]	<element name="contributor"> <element name="corporatename"> <element name="spa"> <field name="value">	
Año (opcional)	260|c 264|c	dc.date.issued[spa] dc.date.issued	<element name="date"> <element name="issued"> <field name="value">	Código 'lsr04'
	773|y	dc.relation.conferencedate[spa] dc.relation.conferencedate	<element name="relation"> <element name="conferencedate"> <element name="spa"> <field name="value">	
Título del documento (opcional)	245|a 245|b	dc tittle[spa] dc tittle[eng]	<element name="title"> <element name="spa"> <field name="value"> <element name="eng"> <field name="value">	Código 'lsr05'
Tipo de material (manual, cartilla…) (opcional)	521|3	dc.type.local[spa] dc.type.local	<element name="type"> <element name="local"> <element name="spa"> <field name="value">	Código 'lsr06'
Título del proyecto de investigación (opcional)	590 |a	dc.relation.project[spa] dc.relation.project	<element name="relation"> <element name="project"> <element name="spa"> <field name="value">	Código 'lsr07'
Código del proyecto SIEMBRA (opcional)	536 |d	dc.relation.codproject[spa] dc.relation.codproject	<element name="relation"> <element name="codproject"> <element name="spa"> <field name="value">	Código 'lsr08'
Código local del proyecto (opcional)	536 |f	dc.relation.codlocalproject[spa] dc.relation.codlocalproject	<element name="relation"> <element name="codlocalproject"> <element name="spa"> <field name="value">	Código 'lsr09'
Instituciones aliadas (opcional)	710 2# |a	dc.contributor.corporatename[spa] dc.contributor.corporatename	<element name="contributor"> <element name="corporatename"> <element name="spa"> <field name="value">	Código 'lsr10'

	911 |a	dc.description.sponsorship [spa] dc.description.sponsorship	<element name="description"> <element name="sponsorship"> <element name="spa"> <field name="value">	
Nombre del Sistema de información de los proyectos de la entidad (opcional)	500| a	dc.relation.siproject[spa] dc.relation.siproject[	<element name="relation"> <element name="siproject"> <element name="spa"> <field name="value">	Código 'lsr11'
Título de la oferta tecnológica (Opcional)	591 |a	dc.relation.offer[spa] dc.relation.offer	<element name="relation"> <element name="offer"> <element name="spa"> <field name="value">	Código 'lsr12'
Código de la oferta en SIEMBRA (Recomendación, Innovación en la Mecanización) (Opcional)	592 |a	dc.relation.codsowingoffer[spa] dc.relation.codsowingoffer	<element name="relation"> <element name="codsowingoffer"> <element name="spa"> <field name="value">	Código 'lsr13'
Código de la oferta local (Opcional)	593 |a	dc.relation.codlocaloffer[spa] dc.relation.codlocaloffer	<element name="relation"> <element name="codsowingoffer"> <element name="spa"> <field name="value">	Código 'lsr14'
Aceso Referencial En texto completo	856 |u	dc.identifier.uri	<element name="identifier"> <element name="uri"> <field name="value">	Código 'lsr15'

Para ello es necesario tener en cuenta las tipologías para la recuperación de la información:

1 Resolución

2 Ley

3 Decreto

4 Bac

5 Planes

6 Convenio

7 Libro

8 Artículo científico

9 Marco conceptual y operativo

10 Manual

11 Estudio

12 Guía

13 Página web

14 Informes

15 Capitulo

16 Boletín

17 Conpes

18 Cartilla

19 Acuerdo

20 Fichas socioeconómicas

21 Referentes internacionales

22 Estudios de vigilancia

23 Fichas de contexto

24 Documento de trabajo

25 Modelo Productivo

26 Multimedia

27 Póster

28 Informes

29 Tesis

30 Trabajos de grado

31 Norma

Posteriormente en el año 2023 se realizó el cambio de versión de Primo a Primo VE, por lo cual se requiere de la configuración de una API de parte del DTI junto con el proveedor y la BAC, donde se debe tener en cuenta los siguientes parámetros de consulta:

Los filtros que se deben tener en cuenta, son: Año de publicación, Departamento, Tipología

Así las cosas, se crean los siguientes campos en Alma y Primo para la recuperación de las facetas y configuración en Siembra. Estos son:

- Departamento: discovery.local1
- Sistemas productivos: discovery.local3
- Recurso Local: dc:type
- Miniatura thumbnail : dcterms:relation

Para realizar dichas búsquedas, en el ambiente de pruebas de Siembra: http://siembrad2.corpoica.org.co:8081/HomeNew , se debe realizar las búsquedas de la siguiente manera:

Búsqueda por el filtro departamento es: Configurado bajo el código lds05

https://api-na.hosted.exlibrisgroup.com/primo/v1/search?&vid=57BAC_INST:BAC&scope=RI_BAC&q=any,contains,papa&apikey=l8xx066955af8e434edb84fc0fe545a1c1c6&limit=10&offset=0&sort=date_d&qInclude=facet=lds05,include,Cundinamarca

Búsqueda sobre Fecha de creación: Configurado bajo la búsqueda include ####
Campo: creationdate

https://api-na.hosted.exlibrisgroup.com/primo/v1/search?&vid=57BAC_INST:BAC&scope=RI_BAC&q=any,contains,papa&apikey=l8xx066955af8e434edb84fc0fe545a1c1c6&limit=10&offset=0&sort=date_d&qInclude=creationdate,include,2005

Búsqueda en Primo:
https://agrosavia.primo.exlibrisgroup.com/discovery/search?query=any,contains,Papa&tab=ALL&search_scope=RI_BAC&vid=57BAC_INST:BAC&facet=searchcreationdate,include,2005%7C,%7C2005&lang=es&offset=0

Búsqueda por filtro Entidad (Publisher):

https://api-na.hosted.exlibrisgroup.com/primo/v1/search?&vid=57BAC_INST:BAC&scope=RI_BAC&q=any,contains,papa&apikey=l8xx066955af8e434edb84fc0fe545a1c1c6&limit=10&offset=0&sort=date_d&qpublisher,include,Corporaci%C3%B3n%20colombiana%20de%20investigaci%C3%B3n%20agropecuaria%20-%20AGROSAVIA

Audiencia: Parcela extensionistas:
Se debe realizar bajo el campo dc.audience en dspace, configurada la búsqueda bajo el código lds08

ANOTACIONES:

1. AGROSAVIA junto con el proveedor de Primo, se encargarán de realizar la normalización y configuración de los metadatos para la API de consumo y adaptar el diseño de acuerdo con el requerimiento
2. En cuanto a la visualización de la información en la API se deben visualizar los siguientes campos: autor, título, tipo de material, año, miniatura del recurso, URL de acceso al recurso, departamento.
3. La API que entregará el proveedor junto con el DTI debe estar funcionando con los parámetros establecidos y de acuerdo con las indicaciones dadas en este documento.
4. El proveedor debe dar el soporte técnico necesario durante todo el proceso de implementación y puesta en marcha de la API
5. En el manual de usuarios entregado por el proveedor deben estar explícitos la forma de enviar cada uno de los parámetros para consumir el servicio.

Limitación en el número de resultados:

Cuando se consulta Primo Central a través de la API, que se limitan a la solicitud de registros dentro de los primeros 200 resultados. Es decir, si está recuperando

registros con el fin de mostrar 10 registros por página, sólo puede mostrar las primeras 20 páginas. Cualquier solicitud que se envía para recuperar los registros más allá de los primeros 200 registros dará lugar a un conjunto de resultados vacío.

Firmado digitalmente por Victor Manuel Mondragon Maca