UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

# Extracción automática de información sobre movimientos socio-territoriales rurales en Uruguay

Informe de Trabajo Final de Ingenieria presentado por

Nicolás Inzua Herken, Tomás Rodríguez Dallo

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingenieria en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisor

Juan José Prada

Montevideo, Octubre 2025

# Agradecimientos

Agradecemos profundamente a nuestras familias por su apoyo incondicional, a nuestros tutores por su guía y a nuestros amigos y compañeros de carrera por su constante apoyo durante este proceso. A todos quienes contribuyeron, directa o indirectamente, nuestro sincero reconocimiento.

**Nicolás Inzua**
**Tomás Rodríguez**

# Resumen

Este proyecto desarrolla un sistema automático de extracción de información estructurada a partir de noticias relacionadas con movimientos socioterritoriales rurales en Uruguay. Utilizando Modelos de Lenguaje de Gran Escala (LLMs) y técnicas de Procesamiento de Lenguaje Natural (PLN), el sistema procesa alertas de Google, extrae información relevante y la estructura según las necesidades del Observatorio de la Cuestión Agraria del Uruguay (OCAU). La solución implementa un *pipeline* de procesamiento que incluye clasificación de relevancia, extracción paralela y secuencial de información, y validación mediante esquemas estructurados con Pydantic. El sistema integra técnicas de procesamiento de lenguaje natural (PLN) con modelos de lenguaje de gran escala (LLMs) para identificar, clasificar y analizar las acciones de movimientos sociales y su impacto geográfico. La arquitectura propuesta incorpora tecnologías innovadoras como Model Context Protocol (MCP) para la modularización de servicios de IA y Pinecone como base de datos vectorial para la clasificación semántica de relevancia. El sistema logra automatizar un proceso que anteriormente requería análisis manual intensivo, proporcionando una herramienta valiosa para el Observatorio de la Cuestión Agraria del Uruguay (OCAU) en su labor de monitoreo y análisis de la dinámica socioterritorial rural.

**Palabras clave:** Procesamiento del Lenguaje Natural, Modelos de Lenguaje de Gran Escala, Extracción de Información, Análisis de Noticias, Reconocimiento de Entidades Nombradas.

# Índice general

# Capítulo 1

# Introducción

Los movimientos socioterritoriales rurales constituyen un actor clave para comprender las transformaciones sociales, políticas y económicas del Uruguay contemporáneo. El Observatorio de la Cuestión Agraria del Uruguay (OCAU) realiza un seguimiento sistemático de estas dinámicas a partir de fuentes periodísticas y comunicados institucionales. Sin embargo, el proceso manual actualmente empleado —que incluye la revisión diaria de alertas, el filtrado de relevancia y la carga de información en formularios estructurados— presenta limitaciones de escalabilidad, consistencia y trazabilidad.

Este trabajo propone el diseño e implementación de un sistema automatizado de extracción de información que convierte texto no estructurado (noticias y comunicados) en datos estructurados alineados con los modelos del OCAU. La solución integra módulos de ingesta, clasificación temática, extracción asistida por modelos de lenguaje de gran escala (LLMs), validación de esquema y persistencia. Además, se establecen criterios de evaluación y métricas que permiten medir el desempeño del *pipeline* en términos de cobertura, precisión y consistencia.

A continuación se presenta el contexto del problema, las fuentes y conceptos que fundamentan el enfoque, y los objetivos que guían el desarrollo del sistema.

## 1.1. Contexto y motivación

El Observatorio de la Cuestión Agraria del Uruguay (OCAU), conformado por docentes de diferentes servicios de la Universidad de la República, realiza un seguimiento sistemático de estos movimientos a través del análisis de noticias y eventos relacionados con el ámbito rural y agrario. Actualmente, este proceso de monitoreo y análisis se realiza de forma manual, requiriendo que los investigadores:

- Revisen diariamente las alertas de Google configuradas con palabras clave específicas
- Filtren manualmente las noticias relevantes
- Extraigan información estructurada de cada noticia
- Completen formularios con múltiples campos categorizados

Este proceso manual presenta limitaciones significativas en términos de escalabilidad, consistencia y eficiencia temporal, motivando el desarrollo de una solución automatizada.

## 1.2. Noticias y eventos

Las noticias periodísticas y comunicados institucionales constituyen la fuente principal de insumos para este trabajo, ya que contienen descripciones ricas en entidades, eventos y relaciones que son objeto de extracción automática.

A modo de ejemplo, el siguiente fragmento de la noticia publicada por El Observador en el año 2023 introduce el contexto de reclamos institucionales en el sector agropecuario. En negrita se destacan las organizaciones protagonistas, elementos clave que hacen relevante esta noticia para el dominio de estudio:

> *"Dos de las principales gremiales del sector agropecuario, la* ***Asociación Rural del Uruguay (ARU)*** *y la* ***Federación Rural (FR)*** *—integrantes de Campo Unido— solicitaron al Poder Ejecutivo una reducción de la alícuota de la Contribución Inmobiliaria Rural para los propietarios de padrones rurales, considerando que persisten las dificultades generadas por la sequía."*

Este fragmento muestra un ejemplo de noticia real. El contenido completo de la noticia puede consultarse en el Anexo 7.3.

## 1.3. Práctica vigente: monitoreo manual con alertas de Google

Actualmente, el seguimiento de noticias relevantes para el estudio de movimientos socioterritoriales se realiza íntegramente de forma manual por parte del equipo de investigación del OCAU. Este proceso, comprende un flujo de trabajo estructurado en las siguientes etapas:

1. **Configuración de alertas**: definición de un conjunto especializado de palabras clave que incluyen nombres de movimientos específicos (SIPES, CNFR, Red de Agroecología del Uruguay, Un Solo Uruguay, etc.), acciones características (manifestaciones, paros, movilizaciones) y temáticas sectoriales (políticas públicas agrarias, conflictos ambientales, cuestión territorial) en Google Alerts y suscripciones a boletines institucionales.

2. **Revisión semanal rotativa**: distribución de la carga de trabajo entre los integrantes del equipo, donde cada investigador se responsabiliza de revisar las alertas recibidas durante una semana específica del mes, realizando la lectura completa de cada alerta, apertura de enlaces y descarte preliminar de contenido irrelevante.

3. **Filtrado de relevancia**: aplicación de criterios metodológicos especializados para determinar la pertinencia de cada noticia según su relación con movimientos sociales rurales, políticas públicas agrarias, o conflictos ambientales y territoriales en Uruguay

4. **Registro estructurado**: registro estructurado en formulario *Jotform*[1] con campos categorizados que incluyen referencia cartográfica, tipología de movimientos, instituciones involucradas, acciones del estado, objetivos de las acciones, y clasificación en acciones matrices y derivadas. El dominio completo de estos campos de extracción se detalla en el Anexo 7.3.

[1] Plataforma en línea para la creación de formularios y encuestas. Documentación oficial: https://www.jotform.com/help/

5. **Codificación detallada**: completado manual de múltiples dimensiones analíticas para cada noticia, incluyendo georreferenciación a nivel región-departamento-localidad, clasificación de tipos de movimiento (agricultura familiar-campesina, trabajadores rurales asalariados, etc.), identificación de instituciones estatales y acciones gubernamentales, y categorización según objetivos de desarrollo sostenible.

**Limitaciones del enfoque actual.**

- **Escalabilidad limitada**: el volumen de alertas fluctúa y crece; la revisión manual consume tiempo y no escala linealmente con el equipo.
- **Consistencia variable**: la interpretación manual de criterios metodológicos puede introducir variabilidad entre codificadores y sesgos temporales, especialmente en la clasificación de objetivos de acciones (propositivas, resistencia, defensiva, reivindicativa, emancipatorios) o la identificación de acciones derivadas (marcha/paro, proyectos, eventos, etc.), bajo la definición de dominio del OCAU.
- **Trazabilidad incompleta**: resulta costoso reconstruir decisiones (por qué se aceptó/rechazó un ítem, qué versión del criterio se usó).
- **Riesgo de duplicados y omisiones**: enlaces repetidos, re-publicaciones o notas similares pueden registrarse de forma inconsistente; algunas noticias relevantes pueden pasar desapercibidas.
- **Latencia en el análisis**: la dependencia de ventanas de revisión manual y la necesidad de completar manualmente formularios extensos introduce retrasos significativos entre la publicación de una noticia y su disponibilidad para análisis.

**Objetivo del proyecto: automatización inteligente como optimización estratégica**

Es importante destacar que la problemática descrita **no constituye una crisis operacional del OCAU**, sino una oportunidad de optimización estratégica. El trabajo manual actual ha demostrado su valor científico y ha permitido la construcción de una metodología robusta y *datasets* de calidad. Sin embargo, la incorporación de herramientas de automatización inteligente representa una evolución natural que potenciará significativamente la capacidad analítica del observatorio.

**Contribución del proyecto de automatización**

A partir de este diagnóstico, el proyecto apunta a:

- **Automatización de la ingesta de datos**: consolidación automática de múltiples fuentes de información y normalización de contenidos, reduciendo significativamente el tiempo dedicado a tareas de recopilación y organización preliminar.
- **Clasificación asistida de relevancia**: desarrollo de un sistema que pre-filtra automáticamente las noticias según los criterios metodológicos del OCAU.
- **Extracción estructurada de información**: automatización de la identificación y codificación de elementos clave como ubicación geográfica, actores institucionales, tipos de movimientos y acciones, aplicando las categorías y criterios de clasificación desarrollados por el OCAU.

- **Automatización de carga en plataformas web**: integración automática de los resultados procesados hacia plataformas web especializadas donde se pueden visualizar y analizar los datos extraídos, eliminando la necesidad de carga manual en estos sistemas.

## 1.4. Objetivos del proyecto

Este proyecto busca transformar noticias sobre movimientos socioterritoriales rurales en Uruguay en datos estructurados útiles para análisis y toma de decisiones por parte del OCAU. A continuación se definen el objetivo general y los objetivos específicos, redactados de forma operacional y con criterios de éxito asociados.

### 1.4.1. El proyecto de movimientos socioterritoriales rurales

Este trabajo se enmarca en un proyecto más amplio de investigación comparada sobre movimientos socioterritoriales rurales en América Latina. El OCAU, como parte de esta iniciativa, ha desarrollado una metodología específica para el caso uruguayo que incluye:

- Un sistema de categorización de movimientos socioterritoriales (11 tipos)
- Una matriz de clasificación de acciones (7 categorías principales con subcategorías)
- Un esquema de referenciación geográfica adaptado al contexto uruguayo
- Criterios de vinculación con los Objetivos de Desarrollo Sostenible (ODS)
- Una tipología de instituciones (9 tipos) y clasificación de acciones del estado (14 tipos de políticas públicas)

El dominio completo de estos campos de extracción, incluyendo todas las categorías, valores posibles y definiciones, se detalla en el Anexo 7.3.

### 1.4.2. Objetivo general

Diseñar e implementar un sistema automatizado de extracción de información que procese noticias sobre movimientos socioterritoriales rurales uruguayos y produzca representaciones estructuradas consistentes con los modelos de datos del OCAU, garantizando escalabilidad y control de calidad mediante técnicas de Procesamiento de Lenguaje Natural y modelos de lenguaje de gran escala.

### 1.4.3. Objetivos específicos

1. **Ingesta de fuentes**: construir un módulo de ingesta automática de Google Alerts.
2. **Filtrado de relevancia**: desarrollar un clasificador para determinar la relevancia de contenidos.
3. **Extracción estructurada**: implementar extractores especializados por dimensión (geográfica, acciones, instituciones, movimientos, ODS), mapeando a los esquemas del OCAU.

4. ***Pipeline* orquestado**: diseñar un *pipeline* reproducible que integre ingesta, clasificación, extracción, validación y persistencia, con registro de eventos y reintentos.

5. **Modelado de datos**: definir modelos que representen fielmente las categorías del OCAU, incluyendo validaciones y *constraints* de integridad.

## 1.5. Estructura del documento

La organización del documento sigue el orden empleado en el trabajo realizado. El capítulo 2 presenta el marco teórico y estado del arte, introduciendo los conceptos fundamentales de PLN, modelos de lenguaje, y las tecnologías empleadas, proporcionando la base conceptual necesaria para comprender las decisiones técnicas posteriores.

El capítulo 3 detalla el análisis del problema y la estrategia empleada, presentando de manera cronológica las tres etapas principales del desarrollo: descubrimiento y diseño inicial, iteración con herramientas MCP y Pinecone, y automatización del proyecto. Aunque el proceso no fue lineal, este capítulo refleja la evolución iterativa que caracterizó el trabajo.

En el capítulo 4 se integran todas las soluciones desarrolladas, presentando la arquitectura final del sistema con sus módulos principales y el *pipeline* completo que los conecta. El capítulo 5 presenta el análisis de resultados mediante evaluación sobre documentos reales, incluyendo una discusión sobre las limitaciones identificadas.

Finalmente, el capítulo 6 expone las métricas operativas y análisis de costos, mientras que en el capítulo 7 se presentan las conclusiones y perspectivas de trabajo futuro. Esta estructura permite una comprensión progresiva del desarrollo del sistema, desde los fundamentos teóricos hasta la evaluación crítica de los resultados obtenidos.

## 1.6. Cronograma de trabajo

El desarrollo del proyecto se extendió desde septiembre de 2024 hasta octubre de 2025, estructurándose en etapas secuenciales que respondieron a la naturaleza iterativa del trabajo. La Tabla 1.1 presenta la distribución temporal de las principales fases del proyecto.

| **Etapa** | **Duración** | **Fecha de inicio** |
|---|---|---|
| Comprensión del problema e investigación de trabajos relacionados | 2 meses | Septiembre 2024 |
| Etapa 1: Descubrimiento y diseño de la funcionalidad inicial | 3 meses | Noviembre 2024 |
| Etapa 2: Iteración sobre extractores, pipelines, MCPs y Pinecone | 4 meses | Febrero 2025 |
| Etapa 3: Automatización del proyecto | 2 meses | Junio 2025 |
| Redacción y corrección del informe final | 3 meses | Julio 2025 |
| Preparación de la defensa | 2 semanas | Octubre 2025 |

Tabla 1.1: Etapas del proyecto con duración y fecha de inicio

# Capítulo 2

# Marco teórico y estado del arte

En este capítulo se describen los principales conceptos, herramientas y enfoques técnicos utilizados en el desarrollo del proyecto. Se presentan de forma concisa y ordenada las tecnologías relevantes en el área de Procesamiento del Lenguaje Natural, los modelos de lenguaje de gran escala, las arquitecturas de extracción de información estructurada, y los mecanismos de validación de datos.

También se introducen formalmente las bibliotecas y marcos tecnológicos empleados en la implementación, incluyendo esquemas de validación con Pydantic, definición de herramientas mediante el protocolo MCP, orquestación de tareas con Celery, y el uso de modelos de lenguaje de OpenAI para la inferencia semántica y la generación de salidas estructuradas.

Finalmente, se revisan trabajos relacionados que abordan la extracción de información estructurada (*structured outputs*) mediante LLMs, validación mediante Pydantic y orquestación con MCP, los cuales sirvieron como fuente de inspiración y referencia técnica para las decisiones de diseño adoptadas en el proyecto.

Este capítulo tiene como objetivo proporcionar una base conceptual clara para el lector, de forma que los términos y tecnologías mencionados en los capítulos posteriores puedan ser comprendidos sin ambigüedades.

## 2.1. Procesamiento de Lenguaje Natural y Modelos de lenguaje

El **Procesamiento de Lenguaje Natural** (PLN) provee los fundamentos para analizar, comprender y generar lenguaje humano, integrando preprocesamiento (tokenización, normalización), representaciones (subpalabras, *embeddings*) y tareas como clasificación, etiquetado de secuencias, extracción de información y generación Jurafsky y Martin (2025).

Los **modelos de lenguaje** aprenden distribuciones sobre secuencias de tokens y permiten predecir o generar texto coherente. En su versión de gran escala (LLMs) —p. ej., BERT Devlin y cols. (2019) y GPT-3 Wei y cols. (2022a)— se han consolidado como estándar. Para extracción estructurada son valiosos porque admiten *salidas con formato* (JSON/clave–valor) y *uso de herramientas externas*, facilitando validación y trazabilidad Bommasani y cols. (2022).

## 2.2. Extracción de información estructurada con modelos de lenguaje

La extracción de información (Information Extraction, IE) es una subdisciplina del PLN enfocada en identificar, clasificar y estructurar datos relevantes a partir de texto libre, transformándolo en representaciones semánticas como entidades, relaciones y eventos que puedan ser procesadas automáticamente Sarawagi (2008); Grishman (2003). Dentro de este marco, la IE clásica se ha formulado como *template filling* (relleno de plantillas) y ha sido evaluada en campañas como la *Message Understanding Conference* (MUC) Grishman y Sundheim (1996); Chinchor y Robinson (1998), habilitando aplicaciones como análisis automatizado y construcción de bases de conocimiento Cowie y Lehnert (1996); Jurafsky y Martin (2025).

De forma general, los **componentes** típicos de IE comprenden:

- **Entidades**: unidades nombradas o conceptuales (p. ej., organizaciones, personas, ubicaciones, artefactos).
- **Relaciones**: vínculos semánticos entre entidades (p. ej., afiliación, pertenencia, *part-of*, *located-in*).
- **Eventos**: ocurrencias ancladas en tiempo y, usualmente, en lugar, que involucran participantes y roles (p. ej., *meetings*, *protests*, *policy enactment*).
- **Atributos**: propiedades de entidades o eventos (p. ej., tipos, objetivos, cantidades, fechas, ubicaciones).

Dentro de esta transformación, la IE permite poblar plantillas (*templates*) con valores extraídos, facilitando tareas como el análisis automatizado, la construcción de bases de conocimiento y el razonamiento estructurado Frederking y cols. (1991).

Tradicionalmente, las técnicas de IE se han basado en **enfoques supervisados**, es decir, métodos que requieren corpus anotados manualmente para aprender a identificar entidades y relaciones. Ejemplos representativos incluyen los *Conditional Random Fields* (CRFs), clasificadores basados en BERT o modelos de secuencia a secuencia entrenados sobre datos etiquetados. Si bien estos métodos han demostrado gran precisión, suelen depender de esquemas fijos, taxonomías predefinidas y una gran cantidad de anotaciones, lo que los hace poco flexibles ante dominios cambiantes o con escasez de datos.

En contraposición, existen **enfoques no supervisados**, que buscan patrones en datos sin anotar mediante técnicas como *clustering*, coocurrencia o medidas estadísticas de asociación. Estos métodos evitan el costo de la anotación manual, pero suelen producir resultados más ruidosos y difíciles de evaluar, ademas de necesitar muchos datos. Entre ambos extremos, los **enfoques semi-supervisados o de *distant supervision*** combinan pequeñas cantidades de datos anotados con grandes corpus sin etiquetas, o bien aprovechan bases de conocimiento para generar etiquetas de manera automática.

Con la irrupción de los LLMs, como GPT-4 o Mistral, ha emergido un paradigma alternativo: la extracción generativa. En este enfoque, las tareas de IE se formulan como problemas de generación textual condicionada, donde el modelo es guiado mediante un prompt que especifica qué información debe devolver. En lugar de clasificar entre opciones predeterminadas, el modelo "describe"los elementos extraíbles en lenguaje natural o en formatos estructurados definidos.

### 2.2.1. De enfoques tradicionales a generativos

Los métodos clásicos de extracción de información —CRFs, clasificadores supervisados o modelos de secuencia a secuencia entrenados en corpus etiquetados— dependen fuertemente de la disponibilidad de datos anotados y de esquemas taxonómicos cerrados. Estas soluciones presentan dificultades para escalar a nuevos dominios, incorporar entidades emergentes, o adaptarse a contextos donde el lenguaje es ambiguo o altamente cambiante.

En contraste, los LLMs permiten reformular la extracción como una tarea de generación textual condicionada, en la que el modelo responde a prompts con salidas estructuradas (por ejemplo, en formato JSON o listas atributo-valor). Este enfoque reduce la necesidad de entrenamiento supervisado y habilita la adaptabilidad a dominios no vistos previamente.

Además, los LLMs son capaces de inferir relaciones no explícitas, identificar entidades fuera del conjunto de entrenamiento y adaptarse a variaciones lingüísticas complejas. Estas capacidades los hacen especialmente adecuados para contextos dinámicos precisamente como los movimientos socioterritoriales, donde las categorías no siempre están bien definidas y el vocabulario evoluciona rápidamente.

### 2.2.2. Capacidades relevantes de los modelos de lenguaje para IE

Una de las principales razones por las que los LLMs han resultado útiles en tareas de extracción de información es su capacidad para comprender, generalizar y estructurar el lenguaje natural en formas útiles para aplicaciones posteriores. A diferencia de los enfoques supervisados clásicos, que dependen de esquemas fijos y conjuntos de entrenamiento balanceados, los LLMs permiten aplicar extracción en dominios abiertos y contextos dinámicos.

Algunas de las capacidades más relevantes incluyen:

- **Comprensión contextual profunda:** los LLMs pueden desambiguar entidades y relaciones teniendo en cuenta el contexto semántico y discursivo, más allá de patrones superficiales Devlin y cols. (2019); Rogers y cols. (2020).
- **Adaptabilidad a nuevos dominios:** gracias al aprendizaje generalista e instruccional, muestran generalización *zero-/few-shot* y rápida adaptación a tareas o dominios no vistos Wei y cols. (2022a,b); Bommasani y cols. (2022).
- **Inferencia implícita:** son capaces de recuperar conocimiento latente y realizar razonamientos de forma implícita, extendiendo la extracción más allá de la coincidencia directa Petroni y cols. (2019); Kojima y cols. (2023).
- **Tolerancia a variabilidad lingüística:** operan sobre múltiples estilos, registros y lenguas, manteniendo consistencia estructural en las salidas Conneau y cols. (2020); Xue y cols. (2021).

### 2.2.3. Representación estructurada de salidas

Una de las transformaciones más significativas que introducen los modelos de lenguaje en la tarea de extracción de información es la posibilidad de generar directamente salidas con estructura semántica explícita, como objetos JSON o pares clave–valor. En lugar de identificar etiquetas dentro de una secuencia (en el enfoque clásico de etiquetado BIO[1]),

[1] El esquema BIO (Begin, Inside, Outside) es un método estándar para etiquetado de secuencias en PLN. Véase Ramshaw y Marcus (1999) para su formulación clásica basada en *chunking* y Tjong Kim Sang y Buchholz (2000) para su adopción y evaluación en el shared task de CoNLL-2000.

los LLMs pueden ser instruidos para construir representaciones que codifican entidades, relaciones o eventos de forma organizada.

Estas salidas estructuradas presentan múltiples ventajas:

- **Facilitan la integración con sistemas posteriores**, al eliminar la necesidad de pasos intermedios de postprocesamiento.
- **Permiten validación automática**, ya que pueden ajustarse a esquemas como JSON Schema o modelos tipados.
- **Mejoran la trazabilidad y la auditoría**, al explicitar cada campo extraído con su correspondiente valor.

Un ejemplo concreto puede verse en la Figura 2.1, donde el modelo devuelve una representación estructurada de un evento, indicando explícitamente la acción, la ubicación y la fecha correspondiente.

```
{
  "evento": "protesta",
  "ubicacion": "Tacuarembó",
  "fecha": "2025-07-15"
}
```

Figura 2.1: Ejemplo de salida estructurada en formato JSON generada por un LLM.

## 2.3. Técnicas de prompting para extracción de información

Los LLMs no requieren reentrenamiento para adaptarse a nuevas tareas, sino que pueden ser condicionados mediante prompts cuidadosamente diseñados. Esta propiedad ha transformado la forma en que se aborda la extracción de información, permitiendo formular la tarea en términos puramente textuales.

Estudios recientes señalan que incluso cambios mínimos en la redacción o formato del *prompt* pueden causar variaciones dramáticas en la calidad de la salida. Por ejemplo, en entornos *few-shot*—que se define más adelante—se ha observado que ajustes casi imperceptibles en el formato del prompt pueden generar diferencias sustanciales en el rendimiento de los modelos Sclar y cols. (2024).

De forma similar, se ha demostrado que modificaciones tan triviales como añadir un espacio al final del prompt pueden alterar por completo la respuesta de un LLM Salinas y Morstatter (2024).

Por estas razones, se han desarrollado estrategias sistemáticas de *prompt engineering*, diseñadas para explotar al máximo las capacidades latentes de los modelos incluso sin necesidad de entrenamiento adicional.

En este contexto, el *prompting* consiste en redactar instrucciones, ejemplos o formatos deseados para guiar al modelo en la generación de salidas estructuradas. Existen varias

estrategias según el grado de supervisión o estructura implicada, que se comentan en las siguientes subsecciones:

### 2.3.1. Zero-shot, one-shot y few-shot prompting

Los modelos de lenguaje de gran escala pueden ser instruidos para resolver tareas específicas mediante el diseño cuidadoso de *prompts*. Dependiendo de la cantidad de ejemplos que se incluyan en la instrucción, se distinguen varias modalidades:

- **Zero-shot prompting:** se le pide al modelo que realice una tarea directamente a partir de una descripción textual, sin ejemplos adicionales. Por ejemplo: *"Extrae la entidad geográfica y el tipo de acción del siguiente párrafo: [...]"*.
- **One-shot prompting:** se proporciona un único ejemplo que ilustra el comportamiento esperado, seguido de una nueva instancia sobre la cual el modelo debe operar. Este enfoque suele ser útil cuando la tarea requiere mostrar el formato de salida esperado, pero no se dispone de múltiples ejemplos.
- **Few-shot prompting:** se incluyen varios ejemplos dentro del *prompt*, los cuales ilustran la tarea y permiten al modelo inducir patrones a partir de esos casos. Posteriormente, esos patrones son aplicados a nuevas instancias.

Estas estrategias aprovechan las capacidades inductivas de los LLMs y permiten adaptarse a nuevos dominios sin necesidad de un entrenamiento adicional.

En el proyecto se optó por un enfoque de **few-shot prompting**, ya que la disponibilidad de ejemplos representativos permitió guiar al modelo de manera más precisa hacia la estructura de salida deseada, reduciendo la ambigüedad y aumentando la consistencia en la extracción de información.

### 2.3.2. Prompting de tipo Chain-of-Thought (CoT)

El *Chain-of-Thought prompting* consiste en inducir al modelo a razonar paso a paso antes de entregar una respuesta final. Esta técnica ha demostrado mejorar el rendimiento en tareas que requieren lógica, inferencia o interpretación contextual. Aunque su uso se ha popularizado especialmente en tareas de razonamiento lógico y matemático, también puede aplicarse a la extracción de información compleja, donde es necesario descomponer o justificar la inferencia.

Por ejemplo, se puede inducir al modelo a:

1. Identificar las entidades presentes.
2. Inferir la relación entre ellas.
3. Generar la salida estructurada final.

### 2.3.3. Consideraciones para prompting orientado a estructura

Cuando el objetivo es obtener salidas en un formato específico (por ejemplo, JSON válido o estructuras compatibles con modelos tipados), el prompt debe incluir no solo la tarea, sino también la forma de la respuesta esperada.

Algunas buenas prácticas incluyen:

- Indicar explícitamente la estructura requerida (por ejemplo: *"Devolvé un objeto JSON con las claves* `ubicación`, `evento` *y* `fecha`*"*).

- Incluir un ejemplo del formato deseado.

Estas prácticas ayudan a mejorar la consistencia de las respuestas y facilitan su validación automática.

## 2.4. Validación estructural de salidas generadas

Los modelos de lenguaje pueden generar texto con estructura predefinida —por ejemplo, en formato JSON, un estándar ampliamente utilizado para representar datos en forma de pares atributo-valor. Sin embargo, cuando se integran modelos generativos en sistemas más grandes, no es suficiente que la salida tenga una estructura aparente: debe cumplir con un esquema esperado que permita su procesamiento automático, sin errores ni ambigüedad.

Para ello, se utilizan mecanismos de validación estructural, que permiten verificar que una salida tenga exactamente los campos requeridos, tipos adecuados y valores consistentes. Esta validación cumple un rol central en tareas como la extracción de información, donde las salidas generadas deben alimentar procesos posteriores, ser almacenadas en bases de datos o cumplir con una interfaz externa.

Una técnica común para esta validación es el uso de esquemas JSON (JSON Schema), un formato estandarizado que describe los requisitos estructurales que debe tener un objeto. No obstante, en entornos de desarrollo en Python, una opción más flexible y legible es el uso de modelos tipados mediante Pydantic.

### 2.4.1. Validación con Pydantic

Pydantic[2] es una biblioteca de Python[3] que permite definir modelos de datos estructurados con validación automática. Estos modelos especifican qué campos se esperan, de qué tipo deben ser y si son opcionales u obligatorios. Al intentar crear una instancia de un modelo Pydantic con datos inválidos, se generará un error detallado explicando la discrepancia.

Esto lo convierte en una herramienta ideal para validar respuestas generadas por LLMs, ya que permite comprobar automáticamente si el modelo respetó el formato requerido.

Como se muestra en la Figura 2.2, Pydantic permite definir un modelo de datos con campos tipados y comprobar automáticamente si una respuesta generada por un LLM respeta la estructura especificada. De este modo, cualquier discrepancia —ya sea por ausencia de campos o por tipos incorrectos— es detectada inmediatamente, antes de que los datos se propaguen a otros componentes.

[2]Documentación oficial de Pydanitc: docs.pydantic.dev.
[3]Python Software Foundation, *Python Documentation*, disponible en `https://docs.python.org/3/`

```python
from pydantic import BaseModel, ValidationError

class ReferenciaCartografica(BaseModel):
    region: str
    departamento: str
    localidad: str

# Caso válido
respuesta_llm = {
    "region": "CENTRO-SUR",
    "departamento": "Florida",
    "localidad": "Berrondo"
}
referencia = ReferenciaCartografica(**respuesta_llm)
print(referencia)

# Caso inválido (localidad debería ser str, no int)
respuesta_llm_invalida = {
    "region": "CENTRO-SUR",
    "departamento": "Florida",
    "localidad": 123
}

try:
    referencia_invalida = ReferenciaCartografica(**respuesta_llm_invalida)
except ValidationError as e:
    print("Error de validación:", e)
```

Figura 2.2: Ejemplo de validación de respuestas con `Pydantic`.

En el ejemplo de la Figura 2.2, si alguno de los campos está ausente o tiene un tipo incompatible (por ejemplo, un número en lugar de una cadena), Pydantic arrojará un error explicativo. Esto permite detectar fallas estructurales inmediatamente, antes de que los datos sean procesados, almacenados o utilizados por otros componentes.

Esta estrategia es ampliamente utilizada en sistemas modernos basados en LLMs, ya que permite tratar a las respuestas del modelo como si fueran datos estructurados formales, brindando robustez y trazabilidad.

## 2.5. Frameworks y arquitecturas para sistemas con LLMs

El uso de LLMs en sistemas reales requiere más que una simple llamada a la API del modelo. Es necesario diseñar flujos de trabajo complejos donde los modelos interactúan con herramientas, validadores, bases de datos y otros servicios. Para ello, han surgido frameworks y arquitecturas especializadas que facilitan la orquestación estructurada de estas operaciones.

En esta sección se describen los principales enfoques y herramientas utilizadas para construir estos sistemas.

### 2.5.1. Construcción de flujos con agentes y herramientas

LangChain[4] es un framework modular diseñado para facilitar la construcción de *pipelines* de procesamiento con modelos de lenguaje. Su propuesta central es la noción de *chain*, una secuencia de pasos donde cada componente puede ser un modelo, una herramienta externa, un prompt o una función de transformación. Esta abstracción permite encapsular lógica compleja en unidades reutilizables y combinables.

Además de las *chains*, LangChain introduce el concepto de herramientas (*tools*), que son funciones parametrizadas que el modelo puede invocar durante la ejecución. Estas herramientas permiten que el modelo interactúe con bases de datos, APIs o validadores, ampliando sus capacidades más allá del texto. Finalmente, LangChain permite definir *agents*, que toman decisiones sobre qué acción ejecutar en cada paso, en función del estado del entorno o de la respuesta previa. A partir de esta definición, en el resto del documento utilizaremos el término agente para referirnos a toda entidad capaz de razonar, decidir y ejecutar acciones dentro de un flujo de procesamiento, siguiendo el enfoque propuesto por LangChain.

- **ReAct (Reasoning and Acting):** combina razonamiento paso a paso con la capacidad de ejecutar acciones externas. Esto permite a los agentes justificar sus decisiones y, al mismo tiempo, actuar sobre el entorno.
- **Conversational Agent:** diseñado para escenarios de diálogo, mantiene contexto conversacional, memoria y un flujo de interacción natural con el usuario.
- **Plan-and-Execute:** separa el proceso en dos fases: primero se genera un plan detallado de pasos a seguir y luego se ejecutan secuencialmente, lo que resulta útil en tareas complejas y de largo horizonte.

LangChain se ha consolidado como una herramienta poderosa para prototipar sistemas conversacionales, tareas de extracción y agentes que combinan razonamiento, memoria y acceso a funciones.

En el ejemplo de la Figura 2.3 se muestra cómo componer una *chain* mínima en LangChain, donde aparecen:

- **Plantilla**: define qué se pide (la consigna) y dónde insertar el texto (`{texto}`).
- **Modelo**: el LLM que responde.
- **Cadena**: encadena pasos para formar el flujo de ejecución.
- **Ejecución**: se entrega el texto de entrada y se obtiene una respuesta en lenguaje natural.

LangChain permite además integrar validadores,—Pydantic en nuestro caso—herramientas externas (*tools*) y conexiones con bases de datos o APIs, habilitando flujos más sofisticados sin aumentar la complejidad conceptual para el lector.

[4]Framework de código abierto para la construcción de aplicaciones con modelos de lenguaje. Permite combinar prompts, modelos y herramientas externas en cadenas de procesamiento modulares. Documentación oficial: langchain.com.

```python
from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser

# 1) Plantilla (prompt): consigna con una variable {texto}
prompt = ChatPromptTemplate.from_template("Resumí en una oración: {texto}")

# 2) Modelo (LLM): intercambiable por variable de entorno (p. ej.,
↪  OPENAI_MODEL)
modelo = ChatOpenAI(model="gpt-4o-mini", temperature=0)

# 3) Cadena: Plantilla -> Modelo -> Parser a texto plano
cadena = prompt | modelo | StrOutputParser()

# 4) Ejecución: se pasa el texto a resumir y se obtiene la salida
salida = cadena.invoke({"texto": "Los reportes del estudio indicaron que..."})
print(salida)
```

Figura 2.3: Cadena mínima en LangChain: *plantilla* → *modelo* → *salida*.

### 2.5.2. Flujos dirigidos con nodos y control de estado

LangGraph[5] es una evolución del enfoque de LangChain orientado al control preciso del flujo de ejecución. A diferencia de las *chains* secuenciales, LangGraph permite modelar el proceso como un grafo dirigido, donde cada nodo representa una operación (por ejemplo, una llamada a un modelo o a una herramienta), y las aristas definen las transiciones condicionales según los resultados parciales.

Esta arquitectura facilita la implementación de flujos de razonamiento más complejos, donde es necesario repetir ciertos pasos, bifurcar rutas según validaciones, o mantener una memoria persistente entre nodos. Además, LangGraph permite almacenar el estado del sistema a lo largo del tiempo, habilitando resumabilidad, trazabilidad y *debugging* fino, lo cual es fundamental para sistemas de extracción que deben mantener coherencia, calidad y eficiencia.

LangGraph puede integrarse con LangChain como motor de ejecución, pero aporta una capa adicional de control y formalismo que lo hace especialmente adecuado para entornos donde la confiabilidad del flujo es crítica. En la Figura 2.4 se muestra un ejemplo de como se construiría un grafo de LangGraph.

[5]Extensión de LangChain para el modelado de flujos de ejecución como grafos dirigidos, donde cada nodo representa una operación (modelo, herramienta o transformación) y las aristas indican transiciones condicionales. Documentación oficial: langchain.com/langgraph.

```
graph = LangGraph()

graph.add_node("extract", extractor)
graph.add_node("validate", validator)
graph.add_node("store", storage)

graph.set_edges("extract", ["validate"])
graph.set_edges("validate", ["store", "extract_if_invalid"])
```

Figura 2.4: Ejemplo de orquestación de flujo con *LangGraph*.

Este tipo de orquestación permite replicar flujos complejos, como los que se usan en sistemas de información, *pipelines* de datos o flujos de extracción multiagente.

### 2.5.3. Hugging Face: acceso a modelos abiertos

Hugging Face[6] complementa los *frameworks* anteriores desde una perspectiva diferente. Más que un motor de orquestación, se trata de un ecosistema integral que centraliza el acceso a modelos de lenguaje abiertos, herramientas de evaluación, datasets y entornos de ejecución.

En el contexto de extracción de información, Hugging Face permite ejecutar modelos como BERT, Mistral y LLaMA2 mediante APIs simples, tanto localmente como en la nube. Sus *pipelines* predefinidos para tareas como clasificación, tokenización o generación permiten resolver rápidamente problemas comunes sin necesidad de diseñar flujos complejos. Esto lo convierte en una herramienta ideal para pruebas rápidas, *benchmarking* o escenarios donde la simplicidad es preferible a la modularidad.

Asimismo, Hugging Face se integra con *frameworks* como LangChain, permitiendo incorporar sus modelos como componentes de un sistema orquestado. Esta interoperabilidad amplía las posibilidades de diseño, habilitando la comparación entre modelos comerciales y modelos *open-source* bajo condiciones controladas.

### 2.5.4. OpenAI: acceso a modelos comerciales y capacidades avanzadas

OpenAI[7] ofrece un ecosistema de LLMs (p. ej., GPT-4, GPT-4o, GPT-4o-mini), que se ha consolidado como una de las principales alternativas comerciales para la integración de LLMs en sistemas de procesamiento. Más que un proveedor aislado, se presenta como un *framework* que incluye mecanismos de interacción, entre ellos la generación de salidas estructuradas en formato `JSON`, lo cual facilita la integración con validadores y flujos de orquestación.

El valor de OpenAI radica en la estabilidad y escalabilidad de sus modelos, así como en la documentación y soporte de ecosistema que lo hacen accesible para la investigación aplicada. Al igual que Hugging Face, se integra con otros *frameworks* como LangChain, permitiendo construir agentes y flujos de extracción estructurada de manera modular.

[6] Documentación oficial de Hugging Face: https://huggingface.co/docs.
[7] Documentación oficial de la plataforma OpenAI: https://platform.openai.com/docs/overview.

## 2.6. Model Context Protocol (MCP)

A medida que los sistemas basados en modelos de lenguaje crecen en complejidad, se vuelve fundamental contar con una arquitectura que permita gestionar el flujo de información, el contexto compartido y la ejecución distribuida de tareas. El Model Context Protocol[8] (MCP) surge como una solución a estos desafíos, proporcionando un esquema organizado para coordinar interacciones entre modelos de lenguaje, herramientas externas, validadores estructurales y demás componentes funcionales de una arquitectura modular.

En lugar de depender de un *pipeline* rígido, MCP introduce un protocolo orientado a mensajes que permite encapsular el estado actual de la tarea, compartir información entre pasos y controlar el recorrido de ejecución. Esta abstracción resulta especialmente útil en entornos asincrónicos, donde distintos módulos (por ejemplo, extractores, herramientas o validadores) deben ejecutarse en distintos tiempos o entornos sin perder la coherencia del flujo.

### 2.6.1. Motivación y fundamentos

En la práctica, los sistemas de extracción de información rara vez operan en una sola llamada de tipo *prompt* → respuesta. Por el contrario, estos sistemas suelen requerir múltiples etapas encadenadas, validaciones intermedias, invocación de funciones externas y mecanismos de control de calidad. Además, el paso de información entre estas etapas necesita una representación clara y coherente del estado.

MCP aborda esta necesidad mediante un objeto de contexto mutable, típicamente representado como un diccionario o modelo tipado, que actúa como contenedor de todos los datos relevantes del proceso: desde el input original hasta los resultados parciales, errores detectados, herramientas invocadas y metadatos auxiliares. Este contexto es leído y actualizado por cada nodo del sistema, lo que permite encadenar operaciones sin acoplar directamente sus implementaciones.

### 2.6.2. Flujo de interacción y control de ejecución

Cada flujo basado en MCP se define como una serie de nodos de procesamiento, donde cada nodo representa una operación que actúa sobre el contexto: por ejemplo, una inferencia con un LLM, una validación estructural, o una llamada a una herramienta externa. Estos nodos están coordinados por un orquestador que determina, en función del estado del contexto, cuál debe ser el siguiente paso.

Esta lógica de control permite definir flujos dinámicos, con bifurcaciones condicionales, reintentos en caso de error o salidas anticipadas si se cumplen ciertos criterios. El resultado es una arquitectura flexible, trazable y extensible.

Un ejemplo simplificado de cómo se representa el contexto en MCP puede verse en la Figura 2.5, donde se incluyen los campos de entrada, los resultados parciales extraídos, las llamadas a herramientas y los posibles errores.

En este caso, el agente logró identificar correctamente la **localidad** (`Paso Severino`), pero no pudo determinar el **departamento** de manera directa a partir del texto de entrada. Ante esta situación, el orquestador invoca la herramienta `map_locality_to_department`, cuyo objetivo es inferir el departamento correspondiente en función de la información geográfica

[8] Protocolo abierto para la coordinación de interacciones entre modelos de lenguaje, herramientas y validadores. Documentación oficial: https://modelcontextprotocol.io/.

disponible en el servidor MCP. De este modo, el flujo combina la capacidad de extracción inicial del modelo con herramientas externas que completan o corrigen la información faltante, asegurando consistencia estructural en el contexto.

```
{
  "input_text": "Vecinos de Paso Severino protestan por la falta de
  ↪  agua.",
  "extracted": {
    "localidad": "Paso Severino",
    "departamento": "Florida"
  },
  "tool_calls": ["map_locality_to_departament"],
  "errors": []
}
```

Figura 2.5: Ejemplo de un contexto MCP en formato JSON con entrada, resultados parciales, llamadas a herramientas y errores.

### 2.6.3. Componentes del protocolo

El protocolo MCP gira en torno a tres componentes fundamentales:

- **El contexto:** Es la unidad central de estado. Puede implementarse como un diccionario o como un modelo estructurado (por ejemplo, con Pydantic) que define los campos esperados, sus tipos, valores por defecto y validaciones.
- **Los nodos de procesamiento:** Son funciones o clases que operan sobre el contexto, realizando tareas como extracción, validación, formateo, invocación de herramientas, selección de rutas, etc. Cada nodo recibe el contexto, lo modifica según su lógica, y lo devuelve al orquestador.
- **El orquestador:** Controla el orden de ejecución de los nodos. Puede implementarse como una máquina de estados, un grafo dirigido (como en LangGraph), o una cola de trabajo distribuida.

La Figura 2.6 muestra un ejemplo de un nodo. Este patrón permite mantener cada operación desacoplada y testeable, mientras el flujo global permanece controlado y observable.

```python
class ExtractionContext(BaseModel):
    input_text: str
    localidad: Optional[str]
    departamento: Optional[str]
    errors: List[str] = []

def extractor_node(context: ExtractionContext):
    context.localidad = extract_localidad(context.input_text)
    return context
```

Figura 2.6: Ejemplo de la implementación un nodo de LangGraph.

## 2.7. Generación Aumentada por Recuperación (RAG)

Los modelos de lenguaje de gran escala (LLMs) presentan limitaciones cuando se enfrentan a tareas que requieren conocimiento externo, específico o actualizado. La técnica de **Generación Aumentada por Recuperación** (Retrieval-Augmented Generation, RAG) surge como una solución a este problema, combinando un módulo de recuperación de información con la capacidad generativa de los LLMs Lewis y cols. (2021).

En este enfoque, el modelo no depende únicamente de lo aprendido en su entrenamiento, sino que puede consultar dinámicamente una base de conocimiento externa. Para ello, la consulta del usuario se transforma en una representación vectorial (*embedding*), la cual se compara contra una colección de documentos indexados. Los fragmentos más relevantes se recuperan y se incorporan al *prompt*, de manera que el modelo de lenguaje genera una respuesta fundamentada en evidencia externa y actualizada.

Esta estrategia permite acceder a información dinámica —como noticias, literatura científica o bases de datos organizacionales— sin necesidad de reentrenar el modelo base. Así, RAG se ha consolidado como una técnica clave en aplicaciones de pregunta–respuesta, asistentes conversacionales y sistemas de soporte a la toma de decisiones.

### 2.7.1. Bases vectoriales

El componente central del proceso de recuperación son las **bases vectoriales**, motores de almacenamiento y búsqueda diseñados para manejar representaciones numéricas de alta dimensión. A diferencia de las bases de datos tradicionales, que indexan información en forma tabular o relacional, las bases vectoriales organizan documentos o fragmentos de texto como vectores en un espacio multidimensional, donde la posición de cada vector refleja características semánticas o léxicas del contenido Johnson y cols. (2017). La búsqueda se realiza mediante medidas de similitud (por ejemplo, distancia coseno, euclídea o producto interno), permitiendo recuperar los elementos más cercanos conceptualmente a una consulta. Además, estas bases soportan metadatos, filtrado y actualización incremental, lo que las hace especialmente adecuadas para escenarios donde el conocimiento evoluciona con rapidez.

### Tipos de representaciones vectoriales

En la literatura y en las implementaciones modernas, pueden distinguirse principalmente dos familias de representaciones vectoriales, cada una con fortalezas y limitaciones que las hacen complementarias.

**Vectores densos (Dense Vectors).** Son representaciones de alta dimensionalidad (típicamente entre 384 y 1536 dimensiones) generadas por modelos de aprendizaje profundo como BERT, GPT o encoders especializados en *embeddings*. Estos vectores buscan capturar el **significado semántico** de los textos, es decir:

- Comprenden el contexto lingüístico de las palabras y frases.
- Detectan relaciones conceptuales, aun cuando no haya coincidencia literal de términos.
- Permiten medir similitud semántica entre documentos que usan sinónimos o paráfrasis.

Por ejemplo, dos noticias que hablen de una "protesta" y de una "manifestación" tenderán a ubicarse próximas en el espacio vectorial, aun si no comparten palabras idénticas.

**Vectores dispersos (Sparse Vectors).** A diferencia de los densos, los vectores dispersos contienen una gran cantidad de ceros y se enfocan en la **presencia explícita de términos** del vocabulario. Procedimientos clásicos como TF-IDF o BM25 Cowie y Lehnert (1996); Jurafsky y Martin (2025) representan cada término como una dimensión y ponderan su frecuencia e importancia en el documento. Más recientemente, algunos motores (por ejemplo, Pinecone) incorporan *embeddings* dispersos entrenados con técnicas modernas, pero que conservan la lógica de coincidencia exacta de palabras. Este tipo de vector es especialmente útil para:

- Detectar nombres propios, siglas o expresiones muy específicas del dominio.
- Mantener precisión terminológica cuando los términos exactos son determinantes para la relevancia.

Por ejemplo, una noticia sobre una "manifestación del PIT-CNT en Tacuarembó" será identificada de inmediato por la coincidencia literal de los términos "PIT-CNT" y "Tacuarembó".

### Representación híbrida y búsqueda combinada

Para superar las limitaciones de cada enfoque por separado, se emplea cada vez con mayor frecuencia una **representación híbrida**, que combina vectores densos y dispersos dentro de un mismo motor de búsqueda. Esta estrategia busca equilibrar dos dimensiones complementarias:

- **Dimensión léxica (*sparse*)**: asegura la detección de términos críticos o técnicos mediante coincidencias exactas.
- **Dimensión semántica (*dense*)**: permite recuperar documentos conceptualmente relacionados, incluso cuando utilizan vocabulario diferente.

La búsqueda híbrida pondera ambas representaciones (mediante un parámetro $\alpha$) para obtener un ranking que maximiza tanto la *precisión* léxica como la *cobertura* semántica.

## 2.8. Ejecución asincrónica

En un flujo sincrónico, cada paso espera a que termine el anterior: si una etapa consulta la red o un servicio externo, todo el proceso queda bloqueado. La ejecución asincrónica separa el **cuándo producir** del **cuándo ejecutar**. La pieza clave es una **cola de trabajo**: el sistema publica tareas en la cola y uno o varios *workers* las van tomando cuando tienen capacidad. Este patrón permite:

- Paralelizar donde conviene.
- Absorber picos con un *buffer* (la cola).
- Reintentar fallas transitorias sin detener el resto.
- Controlar el ritmo para respetar límites de terceros (APIs, LLMs)

### 2.8.1. Componentes típicos

- **Productor**: detecta trabajo nuevo y publica tareas (por ejemplo, cada vez que llega una noticia).
- **Cola (*broker*)**: almacena tareas pendientes hasta que un *worker* las tome.
- ***Workers***: procesos que ejecutan tareas de la cola de manera concurrente.
- **Planificador**: dispara tareas periódicas (por ejemplo, cada N minutos).
- **Almacenamiento y monitoreo**: guarda resultados/estados y permite observar el sistema.

### 2.8.2. Celery: herramienta para tareas distribuidas

**Celery**[9] es un framework de Python para ejecución asincronía basada en colas: define tareas como funciones decoradas, usa un *broker* (Redis o RabbitMQ[10]) para encolar y ejecuta *workers* que consumen y procesan. Provee un **planificador periódico** (Celery Beat) y almacenamiento opcional de resultados. También expone controles operativos clave: **reintentos declarativos** y **rate_limit** por tarea.

#### Componentes y capacidades relevantes

- **Broker**: Redis como cola de mensajes (*buffer* y desacoplamiento).
- **Workers**: procesos concurrentes que ejecutan tareas.
- **Beat (*scheduler*)**: ejecución periódica basado en una configuración estática en horarios.
- **Rutas y colas**: asignar tareas a colas dedicadas para aislar cargas y aplicar cuotas.
- **Monitoreo**: eventos y herramientas para observar estados y latencias.

[9] Documentacion oficial de Celery: https://docs.celeryq.dev
[10] Redis: Documentación oficial. RabbitMQ: Documentación oficial.

## 2.9. Trabajos relacionados

En los últimos años, múltiples iniciativas han explorado la extracción de datos estructurados mediante el uso de modelos de lenguaje de gran escala (LLMs) combinados con tipado y validación estricta. Una de las estrategias más relevantes es el uso de Pydantic como capa de validación de salidas generadas por LLMs, lo cual asegura la consistencia de los datos y facilita su integración en *pipelines* productivos Microsoft Learn (2025c,b).

Un ejemplo representativo es el trabajo que combina *web crawling* con Crawl4AI, extracción semántica con OpenAI y validación de resultados con Pydantic. Este enfoque ha demostrado la viabilidad de orquestar *scraping*, extracción con LLMs y validación tipada en dominios abiertos Gulab (2025); Crawl4AI Project (2025); unclecode (2025). En la misma línea, existen guías técnicas que comparan diferentes estrategias de extracción —basadas en *schemas* o directamente en LLMs— según el nivel de estructura de la fuente Crawl4AI Project (2025).

En entornos empresariales, la documentación de Microsoft/Azure AI ofrece ejemplos de *structured outputs* para la extracción de entidades mediante *function calling*, modelando la salida con clases Pydantic (por ejemplo, Issue o CalendarEvent) y garantizando la calidad de los datos Microsoft Learn (2025c,b,a).

Otra línea de trabajos combina Pydantic con el Model Context Protocol (MCP). Bright Data describe agentes que acceden en tiempo real a la web y devuelven objetos validados Zanini (2025). Por su parte, Box ilustra la generación automática de documentos (p. ej., contratos) a través de un agente que utiliza su servidor MCP y valida las salidas con Pydantic AI Barbosa (2025); Box, Inc. (2025); Novotny (2025).

En conjunto, estos trabajos constituyen un marco técnico consolidado: los LLMs aportan la capacidad de interpretación semántica, Pydantic garantiza la validación y estandarización de datos, y MCP añade interoperabilidad para agentes que requieren orquestación y acceso a fuentes externas. Este paradigma es el antecedente directo de la propuesta de este trabajo en el dominio de las noticias sobre movimientos socioterritoriales rurales.

### 2.9.1. Resumen de aportes relevantes

- **Pipelines scraping→LLM→validación**: viabilidad demostrada con Crawl4AI + OpenAI + Pydantic Gulab (2025); Crawl4AI Project (2025).
- **Structured outputs empresariales**: patrones con Azure/OpenAI que tipan y validan entidades Microsoft Learn (2025c,b).
- **Agentes con MCP**: acceso en tiempo real a datos con contratos Pydantic (Bright Data; Box) Zanini (2025); Barbosa (2025); Box, Inc. (2025).

# Capítulo 3

# Análisis del problema y estrategia empleada

En este capítulo se presenta el análisis del problema y la estrategia de ingeniería adoptada para abordarlo. El objetivo principal es exponer, de manera ordenada, el proceso seguido desde las primeras aproximaciones hasta la consolidación de la solución final.

El desarrollo no fue lineal, sino que se estructuró en una serie de *milestones* que guiaron la evolución del proyecto. Cada una de estas etapas permitió experimentar con diferentes enfoques, identificar limitaciones y tomar decisiones que reorientaron el curso del trabajo. De esta forma, se construyó un camino iterativo de prueba, ajuste y refinamiento.

A lo largo del capítulo se detallarán estos hitos, describiendo las estrategias iniciales, los bloqueos encontrados y las decisiones que llevaron a adoptar soluciones alternativas. El relato se centrará en mostrar cómo cada fase aportó un aprendizaje concreto y cómo dichas experiencias fueron configurando, paso a paso, la arquitectura definitiva del sistema.

Se culminará concluyendo con una síntesis que enlaza naturalmente con el capítulo siguiente, en el cual se expone en detalle la arquitectura resultante de este proceso iterativo.

## 3.1. Etapa 1: descubrimiento y diseño de la funcionalidad inicial

La primera etapa del proyecto tuvo como propósito explorar posibles vías de solución y establecer un prototipo mínimo que sirviera como punto de partida. Se consideró que disponer de un flujo de ejecución completo, aunque básico, resultaba esencial para evaluar la pertinencia de las herramientas seleccionadas y contar con una base sólida sobre la cual avanzar. Esta fase puede entenderse como un *kickstart*, orientado a la familiarización con tecnologías clave y a la validación temprana de conceptos de extracción estructurada mediante modelos de lenguajes.

### 3.1.1. Objetivos y enfoque inicial

En esta etapa se planteó la modularización de la arquitectura en pasos resolubles de manera independiente. La estrategia adoptada consistió en dividir el flujo en tres componentes principales:

1. La obtención de alertas a través de la API de Gmail.
2. El *scraping* y procesamiento de noticias desde las URLs proporcionadas por las alertas
3. La aplicación de modelos de lenguaje para extraer información estructurada definida en esquemas de Pydantic.

Cada componente se desarrolló inicialmente de forma aislada, con el fin de garantizar que cumpliera su función de manera correcta antes de proceder a la integración total. El propósito de esta organización era asegurar que los módulos básicos constituyeran el núcleo de la solución al problema planteado.

### 3.1.2. Implementación de los módulos

#### Ingesta de alertas (Gmail API)

Se implementó un conector que autentica vía OAuth 2.0 y consulta el buzón en búsqueda de Google Alerts. A partir de cada mensaje se obtiene una serie de links que conducen a noticias relevantes para las *keywords* utilizadas por el OCAU. Se registran *timestamps* y metadatos mínimos para trazabilidad, y se manejan errores transitorios con reintentos.

Durante el análisis de los resultados obtenidos mediante este proceso, se identificaron algunas irregularidades en la calidad de las alertas recibidas. Una proporción considerable de falsos positivos se originaba en la detección de palabras clave en secciones del documento HTML ajenas al contenido periodístico principal. Google Alerts identificaba las palabras objetivo en elementos estructurales como *footers*, barras de navegación, secciones de "noticias relacionadas", *sidebars* y otros componentes web, generando alertas para contenidos donde las palabras clave no aparecían en el texto editorial real. Esta problemática evidenciaba la necesidad de implementar una estrategia de procesamiento capaz de distinguir entre menciones genuinas en el contenido periodístico y apariciones incidentales en la estructura del sitio web.

#### *Scraping* de noticias (newspaper3k)

Para abordar las limitaciones identificadas en la ingesta de alertas, se seleccionó *newspaper3k*[1], una biblioteca especializada en la extracción de contenido periodístico que incorpora mecanismos robustos de limpieza de documentos HTML. La biblioteca emplea algoritmos de filtrado que eliminan automáticamente secciones irrelevantes del documento, incluyendo elementos de navegación, *footers*, *sidebars*, bloques de publicidad, secciones de comentarios y contenido relacionado, concentrándose únicamente en la extracción del contenido principal. El módulo descarga y analiza cada URL candidata, extrayendo y normalizando título, autor/es, fecha de publicación, resumen y cuerpo limpio (sin HTML ni elementos estructurales).

#### Extracción estructurada (LLM + LangChain + Pydantic)

La extracción se formuló como generación estructurada: dado el texto de la noticia y el contexto mínimo, el modelo produce un objeto JSON que debe cumplir un esquema Pydantic (tipos, enumeraciones y validaciones). Se utilizó *LangChain* para encadenar pasos (*Runnables*) y gestionar el ciclo *prompt* → respuesta → validación → corrección, con *parsers* que convierten texto a instancias del modelo y reportan errores de esquema. La elección

[1] *newspaper3k*. Documentación: newspaper.readthedocs.io.

inicial de LangChain e integración con modelos hospedados en Hugging Face respondió a la disponibilidad de modelos preentrenados y a la facilidad de iterar *prompts*/componentes; Pydantic aportó el formato de salida y la detección temprana de discrepancias. Se añadieron mapeos y normalización para categorías del dominio (p. ej., geografía, tipos de movimiento, ODS, etc.) y reintentos controlados ante validaciones fallidas.

### 3.1.3. Separación funcional de extractores según el formulario del OCAU

Desde el inicio se modeló la extracción como una secuencia de **cinco bloques** coherentes con las secciones del formulario del OCAU. Cada bloque se resolvió con un **extractor independiente**, ejecutado sobre el mismo LLM pero con **prompts específicos** y **esquemas Pydantic** distintos:

- **Geografía**: incluyendo región, departamento y localidad.
- **Institucional**: incluyendo el tipo de institución, el nombre de la institución y el tipo de acción estado.
- **ODS**: objetivo de desarrollo sostenible.
- **Tipo de movimiento**: incluyendo el tipo de movimiento y el conjunto de nombres de organizaciones presentes.
- **Acciones**: incluyendo el objetivo principal de la acción y la categorización de Acción Matriz y Derivada.

Esta separación permitió iterar y medir cada dimensión por separado, mantener contratos de salida claros y aislar fallos de validación sin comprometer el resto del flujo.

### 3.1.4. Módulo genérico de extracción

Para evitar código repetitivo por cada bloque, se implementó un **módulo genérico de extracción** parametrizable por:

- **Prompt template del bloque**: instrucción y formato esperados para el LLM.
- **Esquema Pydantic objetivo**: contrato de salida con tipos, enumeraciones y validaciones.
- **Reglas de postproceso/normalización**: mapeos y correcciones específicas del dominio.

Usando LangChain como *framework*, se encadenó estos extractores en serie, conformando un *pipeline* ejecutable que avanzaba extractor a extractor.

- **Entrada**: texto normalizado de la noticia (+ metadatos mínimos).
- **Ejecución**: LLM → JSON propuesto → *parseo* a Pydantic → postproceso/normalización.
- **Salida**: instancia válida del esquema del bloque o reporte de errores de validación (para reintento/corrección).

Un esqueleto simplificado del patrón utilizado:

```python
from typing import Generic, TypeVar, Callable
from pydantic import BaseModel, ValidationError

SchemaT = TypeVar("SchemaT", bound=BaseModel)

class GenericExtractor(Generic[SchemaT]):
    def __init__(self, prompt_template: Callable[[str], str], schema:
    ↪  type[SchemaT],
                 postprocess: Callable[[SchemaT], SchemaT] | None = None):
        self.prompt_template = prompt_template
        self.schema = schema
        self.postprocess = postprocess

    def run(self, text: str, llm) -> SchemaT:
        prompt = self.prompt_template(text)
        raw = llm.invoke(prompt)                # respuesta (texto/JSON)
        try:
            obj = self.schema.model_validate_json(raw)  # enforce esquema
        except ValidationError as e:
            raise RuntimeError(f"Schema mismatch: {e}") # se registra para
            ↪  reintento
        return self.postprocess(obj) if self.postprocess else obj
```

La orquestación inicial con *LangChain* se expresó como una *secuencia* de extractores:

```python
from langchain_core.runnables import RunnableSequence

pipeline = RunnableSequence(
    geo_extractor.run
).assign(
    inst=inst_extractor.run
).assign(
    obj=obj_extractor.run
).assign(
    mov=mov_extractor.run
).assign(
    acc=acc_extractor.run
)
# Resultado: diccionario con los cinco bloques validados por Pydantic
```

Esta estrategia permitió:

- **Reutilización del *core* de extracción**: cambiar únicamente el *prompt* y el *schema*.
- **Medición y depuración por bloque**: cada extractor con su propia *suite* de pruebas.
- **Contención de errores**: aislar fallos de un bloque sin arrastrarlos al resto del flujo.

### 3.1.5. Resultados positivos

Se observó que tanto la ingesta de alertas como el *scraping* de noticias alcanzaron un desempeño satisfactorio desde las primeras implementaciones. Estos procesos lograron consolidarse como una base estable y no requirieron mayores modificaciones en etapas posteriores. La estrategia adoptada para estas fases iniciales permitió asegurar que los datos de entrada estuvieran disponibles en forma adecuada para las etapas subsiguientes del *pipeline*.

A los efectos de este trabajo, "satisfactorio" se entiende como el cumplimiento de la siguiente vara operativa:

- **Integridad del texto**: sin pérdidas ni truncamientos relevantes respecto del HTML fuente.
- **Metadatos mínimos válidos**: título, fecha y URL presentes y parseados correctamente.

En cuanto a lo esperado y su verificación, se esperaba recuperar el cuerpo legible de cada noticia (conservando párrafos y sentido) junto con metadatos consistentes para su trazabilidad mínima (título, fecha y URL). La verificación se realizó combinando revisión manual de muestras y compararlas con la pagina web original.

### 3.1.6. Limitaciones identificadas

En contraste, la fase de extracción estructurada presentó dificultades notorias. Se adoptó en primera instancia un enfoque basado en la combinación de Pydantic, HuggingFace y LangChain, inspirado en enfoques descritos en la literatura especializada. No obstante, al aplicarlo al dominio en estudio, se constató que los resultados obtenidos eran inferiores a los esperados.

Se identificó que los casos de éxito documentados correspondían a dominios de menor complejidad, con esquemas más flexibles y *prompts* menos restrictivos. En el presente trabajo, la necesidad de emplear esquemas de Pydantic con enumerados rígidos limitó la capacidad de los modelos para ajustarse a categorías fijas. Esta condición redujo la precisión alcanzada y evidenció los problemas del enfoque inicial para los objetivos del proyecto. Para constatarlo, se ejecutó un proceso de *backtracking*—entendido aquí como la revisión regresiva y reproducible de casos previamente clasificados, comparando la salida del LLM con el esquema objetivo—sobre casos reales ya clasificados por el OCAU: se tomaron noticias y se reejecutó la fase de extracción paso a paso, contrastando cada salida con el esquema esperado. El ejercicio mostró tanto fallas críticas que interrumpían el flujo (errores de parseo y de validación de esquema) como extracciones inconsistentes o carentes de sentido dentro de la ambigüedad propia de un LLM, evidenciando que, en esta configuración, el sistema no alcanzaba el nivel de confiabilidad requerido.

A modo de ejemplo, con entradas *cortas y no ambiguas* el sistema basado en *Pydantic* + LLM lograba mapear correctamente claves geográficas y de acción:

> *"En el noroeste de Uruguay, la asociación de investigadores de Artigas propuso un nuevo proyecto."*

En este caso, pese a la incongruencia leve entre "noroeste" y la mención explícita de **Artigas**, el *pipeline* resolvía la geografía por precedencia del departamento:

- **Geografía esperada**: Región = `NORESTE`, Departamento = `ARTIGAS`.

- **Salida típica (correcta)**: `region=NORESTE`, `departamento=ARTIGAS`.

Sin embargo, al aumentar la **complejidad del contenido** (notas extensas con múltiples actores, siglas y eventos), el rendimiento caía: proliferaban **mapeos incompletos** a enumerados (p. ej., localidades no normalizadas), **siglas sin expandir** (p. ej., `MGAP`) y valores de control como `N/C` por validaciones fallidas de Pydantic. En textos reales, la co-ocurrencia de varias instituciones y movimientos en un mismo artículo incrementó los conflictos de tipificación (actor social vs. organismo estatal) y los *trade-offs* entre etiquetas cercanas (p. ej., marcha vs. manifestación; resistencia vs. reivindicativa).

Para evitar generalizaciones injustas sobre la HuggingFace, se aclara que los resultados aquí reportados corresponden a ensayos con **mistralai/Mistral-7B-Instruct-v0.3** y **v0.2**, además de variantes de **Llama**. Estos modelos mostraron un desempeño adecuado en entradas breves y poco ambiguas (extracción de región/departamento por mención explícita), pero su precisión decreció en documentos largos y no tan explícitos.

Por otro lado, en múltiples oportunidades la etapa de extracción falló porque el LLM no conseguía mapear los valores exigidos por las enumeraciones del esquema —por ejemplo, el catálogo de localidades— a partir del texto de la noticia. Asimismo, la inferencia de siglas y denominaciones institucionales presentó limitaciones: acrónimos como "MGAP" (Ministerio de Ganadería, Agricultura y Pesca) no siempre eran reconocidos, incluso después de ser normalizados, lo que derivaba en validaciones negativas del modelo Pydantic, interrupciones del *pipeline* y, en el mejor de los casos, salidas poco informativas (p. ej., `N/C`, no corresponde).

Por otra parte, se detectaron limitaciones en la etapa de **ingesta** asociadas al uso de *Google Alerts* como disparador del corpus. En numerosos casos, las alertas recuperaron artículos **no relacionados** con las palabras clave definidas o con el dominio de estudio, debido a:

- **Efectos de caché/indexación**: el sistema de alertas prioriza títulos, *snippets* o metadatos previamente cacheados que ya no reflejan fielmente el contenido actual del enlace.

- **Etiquetado SEO engañoso en portales**: algunos sitios asocian palabras clave de alta demanda a notas cuyo cuerpo no guarda relación sustantiva con dichos términos.

Este ruido incrementó la carga de validación manual y afectó la representatividad del conjunto de prueba, condicionando la evaluación del extractor con textos que, aun pasando el umbral de la alerta, no correspondían al fenómeno socioterritorial bajo análisis.

### 3.1.7. Conclusiones de la etapa

Como resultado de esta evaluación, se concluyó que la estrategia inicial debía evolucionar hacia un diseño más modular y flexible, en el cual los extractores pudieran iterarse de manera independiente y se incorporaran mecanismos adicionales de validación. La experiencia de esta primera etapa permitió, por tanto, consolidar una base estable para la ingesta y el *scraping* de datos, al tiempo que puso en evidencia la necesidad de revisar en profundidad el enfoque de extracción estructurada. Estas conclusiones marcaron el inicio de la segunda etapa del proyecto, centrada en la optimización de *pipelines*, la mejora de los extractores y la incorporación de herramientas complementarias.

## 3.2. Etapa 2: iteración sobre extractores, *pipelines*, MCPs y Pinecone

Tras la validación inicial, se consideró necesario reestructurar el enfoque con el fin de mejorar la calidad de extracción y la eficiencia operativa. En esta etapa se priorizó la definición de responsabilidades claras entre módulos, la optimización de la orquestación de los *pipelines* y la incorporación de herramientas que ampliaran las capacidades de los modelos de lenguaje mediante validaciones y recuperación de contexto.

### 3.2.1. Objetivos y enfoque

Se establecieron tres objetivos principales:

- **Mejorar el desempeño de los extractores de forma individual**: apoyarse en una batería de pruebas dedicada para medir y guiar iteraciones.
- **Refinar la secuenciación con *Runnables* (LangChain)**: favorecer el desacople de pasos y la trazabilidad del flujo de ejecución.
- **Incorporar herramientas externas de conocimiento y verificación**: utilizar Model Context Protocol (MCP) para exponer *tools* al agente y búsqueda vectorial con Pinecone para enriquecer el contexto de la noticia a analizar.

### 3.2.2. Estructuración del procesamiento en dos fases

Durante la implementación se observó que **la calidad de las noticias provenientes de Google Alerts presentaba irregularidades**, generando sospechas sobre la legitimidad de su relevancia para el dominio de movimientos socioterritoriales. Google Alerts demostró ser sensible a fallas, capturando frecuentemente noticias que contenían las palabras clave configuradas pero en contextos completamente ajenos al problema de estudio.

A modo de ejemplo, las alertas configuradas sobre la sigla "MDR" (Mesas de Desarrollo Rural, *keyword* de interes para el OCAU), generaron sistemáticamente falsos positivos al abarcar usos ajenos al dominio del estudio. En la práctica, el sistema notificó contenidos sin relación temática (p. ej., notas de ciberseguridad con la técnica *Managed Detection and Response*[2]), lo que contaminó el flujo de recolección y obligó a depurar manualmente los resultados.

A partir de la evidencia, se adoptó una separación explícita del proceso en dos fases: primero, la determinación de relevancia; luego, la extracción. Se observó que filtrar tempranamente las noticias no pertinentes reducía de manera significativa el costo computacional y concentraba los esfuerzos de extracción en los casos de interés. En consecuencia, se diseñó un clasificador de relevancia que combinó OpenAI *embeddings* con un índice vectorial en Pinecone.

[2]Ejemplo de resultado irrelevante capturado por la alerta "MDR": https://www.cronicas.com.uy/sociedad/vigilancia-24-7-el-nuevo-estandar-en-seguridad-ti-para-empresas-exigentes/

### 3.2.3. Incorporación de herramientas MCP y recuperación de contexto

Con el propósito de elevar la precisión, se integró un servidor MCP especializado en geografía, que expuso una herramienta con la documentación de localidades Uruguayas, donde cada entrada contenía el nombre de la localidad, en que departamento se ubicaba, y a que región correspondía. Esta incorporación permitió validar entidades geográficas durante la extracción, mitigando ambigüedades frecuentes del dominio.

Los resultados obtenidos en esta prueba piloto fueron satisfactorios, lo que motivó la decisión de expandir el uso de herramientas MCP. En consecuencia, se optó por desarrollar un servidor MCP con herramientas especializadas disponibles para cada extractor específico, proporcionando así un conjunto de funciones de dominio que complementan las capacidades de los modelos de lenguaje en cada dimensión de extracción.

En paralelo, se habilitó recuperación de contexto a partir de Pinecone, de modo que los extractores pudieran operar con noticias previamente clasificadas y así inferir contexto extra para complementar la información provista por el MCP. Esta recuperación de contexto incluye los resultados de las 5 noticias más similares a la que se está intentando procesar.

### 3.2.4. Optimización de extractores y pruebas

Se procedió a iterar sobre los extractores con un enfoque de mejora incremental. Cada extractor fue sometido a una *suite* de pruebas que permitió cuantificar avances, detectar regresiones y comparar variantes de *prompting* y esquemas de salida. Se constató que la combinación de validaciones MCP con esquemas más estrictos en Pydantic mejoraba la consistencia de los resultados, siempre que se acompañara de recuperación selectiva de contexto y de una secuenciación clara de pasos. En el Capitulo 5 se profundizara en estos resultados.

### 3.2.5. Logros principales y criterios derivados para la arquitectura

De la evaluación de esta etapa se desprendieron criterios de diseño fundamentales que orientaron la arquitectura final:

1. **Separación de responsabilidades mediante clasificación temprana**: La implementación exitosa del módulo de validación de relevancia demostró la necesidad de establecer una etapa de filtrado previo en el *pipeline*. Al descartar contenido irrelevante de manera temprana, se evita el procesamiento innecesario y se reducen significativamente los costos operativos. Esta evidencia establece que la arquitectura debe incorporar clasificación y extracción como componentes distintos, donde la primera actúe como compuerta de acceso que garantice que solo el contenido potencialmente útil llegue a las etapas de análisis posterior.

2. **Desacople y composición modular**: La evolución de los extractores hacia unidades independientes, cada una con responsabilidad única y *prompts* específicos, permite orquestar la ejecución de manera segura y ordenada. Esta modularización facilita el mantenimiento, la depuración y las mejoras iterativas de cada componente por separado.

3. **Validación asistida por conocimiento externo**: La incorporación de herramientas MCP como capa de verificación y normalización de entidades críticas del dominio

demuestra la importancia de enriquecer las extracciones más allá de la información contenida en los *prompts*.

4. **Testabilidad granular**: La necesidad de métricas y pruebas dedicadas para cada módulo de extracción evidencia que la arquitectura debe permitir la evaluación independiente de cada componente.

Estos criterios consolidaron la base conceptual de la arquitectura resultante. La especificación y el diseño detallado de dicha arquitectura se presentan en el Capítulo 4, donde se describen los módulos, sus interfaces y el flujo completo de orquestación.

## 3.3. Etapa 3: automatización del proyecto

Con los componentes de ingesta, clasificación y extracción estabilizados, esta etapa buscó eliminar disparadores manuales y establecer una operación continua sin dependencia de intervención humana. El objetivo era transformar un prototipo funcional en un sistema autónomo capaz de ejecutarse de manera programada, recuperar información de forma fácil, persistir resultados estructurados y proporcionar visibilidad sobre su funcionamiento.

### 3.3.1. Objetivos y alcance de la etapa

Se establecieron cinco objetivos principales que definieron el alcance de esta fase:

- **Automatización del flujo completo**: Eliminar la necesidad de ejecución manual del *pipeline*, estableciendo un ciclo automático que abarque desde la ingesta de alertas hasta la generación de resultados finales.
- **Recuperación programada de información**: Implementar mecanismos que consulten periódicamente las fuentes de datos provenientes de Google Alerts sin intervención humana, garantizando que el sistema capture contenido nuevo de manera continua.
- **Persistencia estructurada de resultados**: Asegurar que cada noticia procesada, independientemente de su resultado (exitoso, inválido o fallido), quede registrada en una base de datos con toda su información estructurada y metadatos de procesamiento.
- **Despliegue operacional**: Configurar el sistema para que pueda ejecutarse de forma continua en un entorno controlado.
- **Visualización y gestión**: Proveer interfaces que permitan monitorear el estado del sistema, consultar resultados procesados y controlar el encendido/apagado de la herramienta según necesidades operativas.

### 3.3.2. Análisis del problema y desafíos identificados

La transición de un prototipo ejecutado manualmente a un sistema autónomo reveló varios desafíos técnicos y operativos que requerían atención específica:

- **Dependencia de servicios externos con límites estrictos**: El sistema depende de proveedores externos que imponen cuotas de uso. La operación continua sin supervisión aumentaba significativamente el riesgo de exceder estos límites, lo que podría resultar en errores de *rate limit*, degradación del servicio o incluso suspensión temporal del acceso.

- **Riesgo de procesamiento duplicado**: Sin controles adecuados, una misma noticia podría procesarse múltiples veces en caso de reintentos o reinicios del sistema, generando registros duplicados y consumo innecesario de recursos.
- **Ausencia de visibilidad operacional**: En un sistema que funciona de manera autónoma, resulta crítico contar con mecanismos que permitan observar su comportamiento, detectar anomalías y comprender el estado actual del procesamiento sin necesidad de inspeccionar *logs* o bases de datos directamente.
- **Programación y coordinación de tareas**: La ingesta de alertas y el procesamiento de noticias requieren ejecutarse en momentos específicos y de manera coordinada, lo que demanda un mecanismo de *scheduling* confiable que gestione la ejecución periódica de tareas.

### 3.3.3. Estrategias de mitigación adoptadas

Para abordar los desafíos identificados, se adoptaron las siguientes estrategias de diseño:

#### Manejo de límites de tasa (*Rate Limits*)

Los proveedores de modelos de lenguaje imponen restricciones sobre el número de solicitudes por minuto (RPM) y *tokens* por minuto (TPM). Cuando estas cuotas se exceden, el servicio responde con errores HTTP 429, interrumpiendo el procesamiento.

Se adoptó una política de reintentos automáticos con ***backoff* exponencial**: cuando una solicitud falla por exceso de tasa, el sistema aplica una espera inicial breve que se duplica progresivamente en cada reintento hasta alcanzar un máximo de intentos. Esta estrategia permite recuperación automática de errores transitorios, adaptación dinámica al estado del servicio, y prevención de sobrecarga mediante esperas crecientes.

#### Ejecución programada y prevención de duplicados

Se estableció un ciclo de ejecución automática cada 4 horas. En cada ciclo, el sistema consulta las Google Alerts a través de la Gmail API solicitando únicamente las alertas recibidas en las últimas 4 horas. Esta ventana temporal sincronizada entre la frecuencia de ejecución y el período de consulta garantiza cobertura completa sin solapamientos, evitando naturalmente el procesamiento duplicado de alertas sin necesidad de mecanismos adicionales de deduplicación.

#### Persistencia categorizada

Se estableció un modelo que registra exhaustivamente todos los documentos procesados, categorizándolos según su resultado: éxitos completos (extracción exitosa), validaciones fallidas (descartados por irrelevancia), errores de *scraping* (contenido no recuperable), y errores de procesamiento (fallos durante extracción). Esta categorización proporciona trazabilidad completa y permite análisis de fallos.

#### Interfaces de observabilidad

Para visibilidad operacional, se diseñaron mecanismos que permiten consultar estadísticas agregadas, inspeccionar noticias individuales con sus metadatos, y controlar el estado del sistema.

### 3.3.4. Transición a la arquitectura consolidada

Las pruebas piloto demostraron operación autosuficiente y confiable del sistema. Esta etapa consolidó principios operativos fundamentales: separación entre coordinación y ejecución, recuperación ante fallos esperables, sincronización temporal para prevenir duplicados, y observabilidad continua del comportamiento del sistema.

Estos principios, se materializan en la arquitectura final presentada en el Capítulo 4, donde se detallan modulos específicos, tecnologías seleccionadas y aspectos técnicos de implementación.

# Capítulo 4

# Arquitectura del sistema e implementación

En este capítulo se expone la implementación final del sistema y la arquitectura de software resultante del proceso iterativo descrito en el capítulo anterior. El propósito es ofrecer una visión integral de cómo se materializó la solución propuesta, detallando las áreas funcionales que conforman el sistema y las tecnologías que las sustentan.

A diferencia de la etapa de análisis y estrategia empleada, aquí se presenta el diseño consolidado: un conjunto de módulos interconectados que, en su conjunto, permiten la ingesta de datos, el procesamiento mediante modelos de lenguaje y la generación de información estructurada de manera confiable y escalable.

El capítulo se organiza en torno a los principales componentes de la plataforma —ingestión, procesamiento, extracción, servicios y análisis—, describiendo sus responsabilidades, interfaces y la forma en que se integran dentro del flujo global. Asimismo, se especifican las herramientas y marcos tecnológicos empleados en cada área (por ejemplo, FastAPI, Celery, Pinecone, OpenAI, MCP y Pydantic), justificando su selección en función de los requisitos del proyecto.

De este modo, se busca no solo documentar la arquitectura lograda, sino también mostrar cómo cada decisión tecnológica responde a las necesidades del dominio, garantizando la solidez, escalabilidad y mantenibilidad del sistema.

## 4.1. Arquitectura general del sistema

La arquitectura del sistema se organiza en cinco módulos principales con responsabilidades bien delimitadas y contratos de entrada/salida explícitos: **Pipeline de Procesamiento**, **Workers de Celery**, **API Backend**, **Frontend** y **Servicios Externos**. Esta descomposición favorece el bajo acoplamiento, permitiendo escalar y evolucionar cada módulo de manera independiente, como se muestra en la Figura. 4.1.

- **Pipeline de Procesamiento**: Núcleo del sistema que ejecuta el flujo secuencial de transformación de datos:
  - **Ingesta de noticias**: Recibe contenidos desde fuentes externas (integración con

*Gmail API*), aplica normalización y produce un documento canónico como insumo para el resto del flujo.

- **Clasificación de contenido**: Decide si un documento avanza en el *pipeline* (relevante) o se descarta (no relevante) según criterios del dominio mediante análisis de relevancia.
- **Extracción con IA**: Transforma documentos relevantes en salidas estructuradas utilizando procesamiento RAG (*Retrieval Augmented Generation*) para extraer información específica del dominio del OCAU (geográfica, acciones, movimientos, institucional, ODS).
- **Generación de resultados**: Consolida las extracciones y prepara los datos para su exportación y persistencia.

- **Workers de Celery**: Gestiona el procesamiento de tareas programadas, ejecutando el pipeline de forma secuencial para cada noticia. Administra la cola de procesamiento mediante Redis como *broker* de mensajes, garantizando reintentos en caso de fallos y manteniendo el orden de ejecución de las etapas del pipeline.
- **API Backend**: Proporciona los *endpoints* REST con autenticación para la interacción con el sistema, gestionando las solicitudes de usuarios y exponiendo los resultados del procesamiento.
- **Frontend**: Interfaz web para visualizar las noticias procesadas, consultar estadísticas básicas del sistema y controlar el encendido o apagado de la herramienta.
- **Servicios Externos y Almacenamiento**:
  - **MongoDB**: Persiste resultados estructurados con metadatos.
  - **JotForm API**: Exportación de datos procesados.
  - **Gmail API**: Fuente de ingesta de noticias.
  - **OpenAI API**: Motor de IA para clasificación y extracción.
  - **Redis**: Gestión de colas y caché para procesamiento asíncrono.

Este marco de referencia guía el resto del capítulo: en las secciones siguientes se profundiza módulo por módulo (Función → Alternativas → Implementación final), y luego se presenta el **pipeline de procesamiento** como mapa de etapas que conecta estas piezas en un flujo único desde la entrada hasta las salidas.

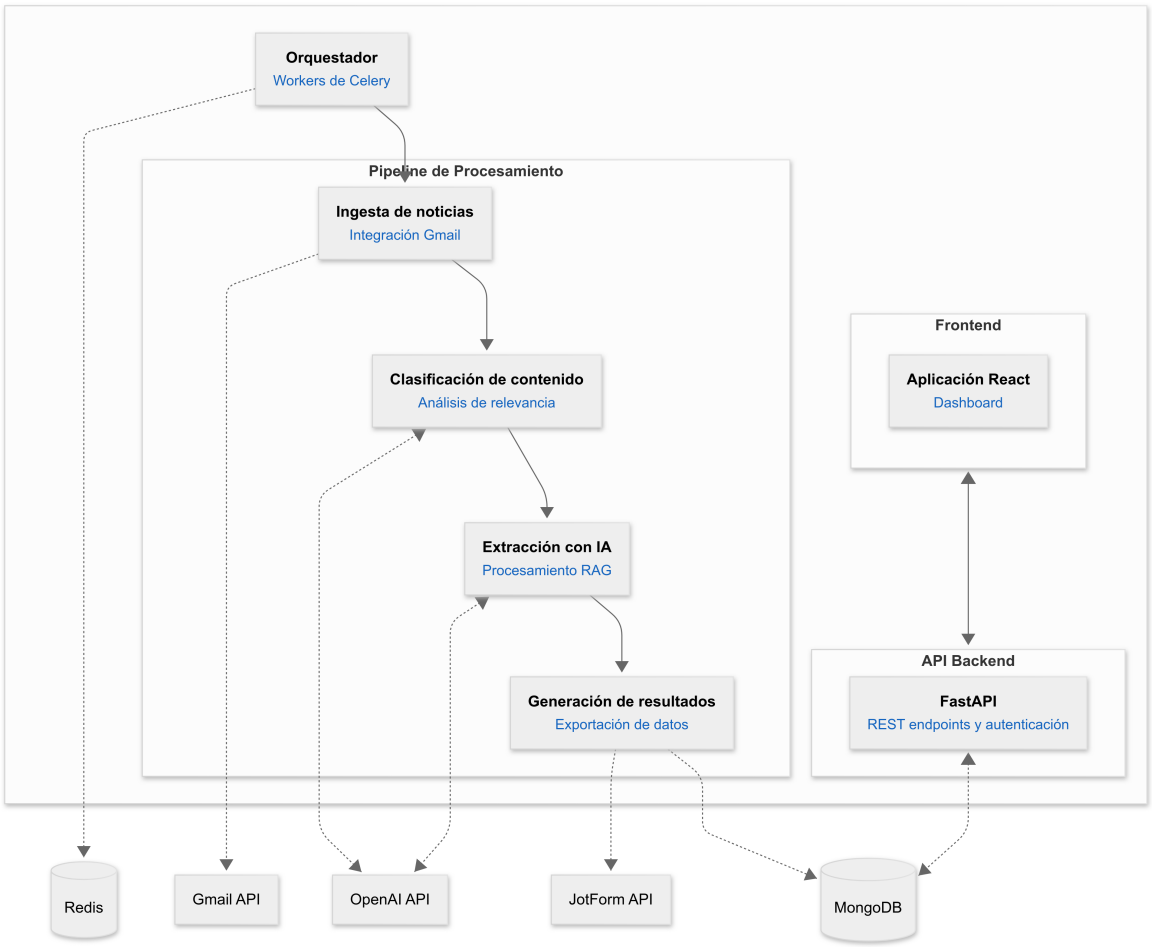



Figura 4.1: Arquitectura general del sistema

## 4.2. Modulo - Ingesta

El módulo de Ingesta constituye el punto de entrada del sistema, responsable de capturar y normalizar contenido periodístico desde fuentes externas. Su función principal es transformar alertas de correo electrónico en documentos estructurados, produciendo un documento canónico que sirve como unidad de procesamiento estandarizada para todo el *pipeline*. Este documento integra tanto los metadatos de origen (fuente, fecha, palabra clave de búsqueda) como el contenido completo del artículo, garantizando que las etapas posteriores trabajen con información consistente y completa. En la arquitectura general presentado en la Figura 4.1, el módulo de Ingesta actúa como la interfaz entre las fuentes externas de información y el **Pipeline de Procesamiento**, estableciendo el contrato de datos que alimenta la etapa de **Clasificación**.

### 4.2.1. Alternativas evaluadas

Como fue mencionado en el capitulo 3, el modulo de Ingesta no recibió ningún cambio significativo desde la etapa 3.1, siendo el prototipo creado en ese momento el mismo utilizado en la arquitectura final.

### 4.2.2. Implementación final

La implementación adoptada combina Gmail API para la obtención de alertas con *newspaper3k* para el *scraping* web.

La integración con Gmail permite recuperar las alertas mediante autenticación OAuth 2.0, extrayendo automáticamente la palabra clave de búsqueda del asunto y las URLs reales del cuerpo del mensaje. Para cada URL identificada, el sistema realiza la extracción del contenido periodístico completo (título, texto, meta-descripción, autores, fecha de publicación), normalizando el contenido para simplificar el procesamiento posterior por los modelos de lenguaje.

La salida del módulo es un documento canónico que combina los metadatos de la alerta con el contenido completo del artículo, listo para la etapa de **Clasificación**. Los fallos durante el *scraping* se registran apropiadamente para mantener la trazabilidad del proceso. Los aspectos de planificación y ejecución distribuida se abordan en la sección de **Orquestación**.

## 4.3. Modulo - Clasificación

El módulo de clasificación constituye un componente crítico del sistema, actuando como filtro de calidad que determina si cada documento de entrada es relevante para el dominio de movimientos socioterritoriales o debe ser descartado. En la arquitectura general (Figura 4.1), este módulo aparece explícitamente como "Clasificación" y actúa como una compuerta de acceso que garantiza que solo el contenido potencialmente útil llegue a las etapas de análisis posterior.

### 4.3.1. Flujo de procesamiento

Cada documento sigue un recorrido en tres pasos que equilibra eficiencia, contexto y precisión.

#### Filtrado preliminar por similitud léxica.

El proceso comienza con una validación difusa de palabras clave implementada con la biblioteca FuzzyWuzzy[1] y el algoritmo `partial_ratio`. Este método calcula la similitud entre subcadenas de texto y asigna un puntaje de 0 a 100; los documentos que superan el umbral configurado (75 %) se consideran potencialmente relevantes. Este primer filtro es rápido y de bajo costo computacional, ideal para descartar noticias que contienen coincidencias superficiales generadas, por ejemplo, por Google Alerts. La estrategia incluye un diccionario que reconoce tanto abreviaturas como denominaciones completas de organizaciones rurales,

[1] FuzzyWuzzy es una biblioteca de código abierto en Python que implementa comparaciones de similitud basadas en la distancia de Levenshtein, pero agrega funciones de normalización, manejo de subcadenas (`partial_ratio`) y puntuación ponderada, lo que la hace más adecuada para comparar fragmentos de texto cortos o no normalizados. Fuente: pypi.org/project/fuzzywuzzy/.

lo que le otorga flexibilidad lingüística y capacidad para manejar variaciones ortográficas o morfológicas.

**Recuperación híbrida de contexto.**

Los documentos que superan el filtro inicial pasan a una búsqueda híbrida en **Pinecone**[2], una base de datos vectorial que permite recuperar información por similitud. La consulta combina dos tipos de representaciones:

- **Vectores densos**, generados con `text-embedding-3-small` de OpenAI, que capturan similitud semántica.
- **Vectores dispersos**, basados en el modelo `pinecone-sparse-english-v0`, que detectan coincidencias léxicas exactas, cruciales para nombres propios y siglas.

**Justificación de la búsqueda híbrida.**

La naturaleza del dominio estudiado —movimientos socioterritoriales rurales en Uruguay— presenta características que justifican una estrategia híbrida:

1. **Nombres propios y siglas poco frecuentes**: Entidades como "USU" (Un Solo Uruguay), "ARU" (Asociación Rural del Uruguay), "MGAP" (Ministerio de Ganadería, Agricultura y Pesca), entre otros, son términos de muy baja frecuencia en corpus generales, pero altamente discriminativos en este contexto específico.
2. **Complementariedad entre enfoques**: Los *embeddings* densos capturan el contexto semántico de las noticias, mientras que los *embeddings* dispersos son excelentes para coincidencias exactas de palabras clave, especialmente importantes en escenarios que involucran terminología específica del dominio y nombres propios (Zhang y cols., 2024).

**Configuración del parámetro $\alpha$.**

Un parámetro $\alpha$ configurable regula la contribución relativa de ambos enfoques en el *score* final de similitud mediante una combinación lineal ponderada. Esta estrategia permite equilibrar dos tipos complementarios de similitud:

- **Similitud conceptual** (vectores densos, peso $\alpha$): Captura relaciones semánticas y temáticas entre documentos, permitiendo recuperar noticias conceptualmente relacionadas aunque no compartan terminología exacta.
- **Similitud léxica** (vectores dispersos, peso $1 - \alpha$): Garantiza recuperación precisa cuando aparecen términos específicos del dominio, especialmente nombres de actores sociales, movimientos, instituciones y siglas características del ámbito rural uruguayo.

En este trabajo se adopta $\alpha = 0{,}7$, valor que privilegia la componente semántica sin perder sensibilidad a coincidencias léxicas ($1 - \alpha = 0{,}3$).

Esta configuración se justifica empíricamente a partir de la Figura 2 del estudio de búsqueda híbrida de Zhang y cols. (2024), donde los tres conjuntos de evaluación (*Natural Questions*, *MS MARCO* y *HotpotQA*) muestran un rendimiento máximo del *Recall@10* en

[2]Documentación oficial: https://docs.pinecone.io

el rango $\alpha \in [0{,}5, 0{,}8]$, con una meseta óptima centrada en $\alpha = 0{,}7$. Este valor equilibra eficazmente la recuperación contextual y la coincidencia literal, y se mantiene robusto en configuraciones sin alineación de puntuaciones, como la utilizada en este proyecto.

Esta ponderación refleja adecuadamente la lógica de búsqueda adoptada en este trabajo: priorizar la identificación de documentos conceptualmente vinculados con un mismo fenómeno socioterritorial rural, sin descuidar las menciones literales de actores, movimientos o instituciones clave. Por ello, se adopta $\alpha = 0{,}7$ como valor óptimo para la ponderación híbrida, alineado tanto con la evidencia experimental del estado del arte como con la naturaleza semántico-léxica del dominio.

La búsqueda devuelve los $k = 5$ documentos más cercanos dentro de un corpus previamente clasificado. La selección de $k = 5$ se fundamenta en prácticas establecidas en sistemas RAG. Lewis y cols. (2021) exploraron sistemáticamente $k \in \{5, 10\}$ en su trabajo fundacional sobre RAG, estableciendo este rango como referencia para equilibrar recuperación de contexto y precisión.

Siguiendo esta recomendación de la literatura, se adoptó $k = 5$ como configuración inicial del sistema. Las pruebas realizadas mostraron resultados satisfactorios en términos de precisión de clasificación, por lo que se adoptó como configuración final del sistema.

**Clasificación final con evidencia contextual.**

Con los documentos similares recuperados, un modelo de lenguaje grande (LLM) realiza la clasificación definitiva. Esta etapa sigue un enfoque *Retrieval-Augmented Generation* (RAG): el LLM evalúa tanto el contenido directo del documento como la evidencia de los textos vecinos, construye un resumen estadístico (puntajes de similitud, títulos y contexto) y emite una decisión binaria de relevancia, acompañada de un puntaje de confianza y un nivel de certeza categórico.

### 4.3.2. Alternativas evaluadas

Durante el diseño se exploraron distintas estrategias, cada una con ventajas y limitaciones que justificaron su descarte en favor de una solución híbrida:

- **Búsqueda Exclusiva por Keywords.** Este enfoque dependía estrictamente de la coincidencia exacta de palabras clave, utilizando un método de clasificación rígido y poco flexible. Al ignorar la similitud semántica, excluía documentos relevantes que no compartieran vocabulario exacto y, al mismo tiempo, introducía falsos positivos por coincidencias superficiales sin contexto. Esta problemática se evidenció empíricamente en las inconsistencias de Google Alerts descritas en la sección 3.2.2, donde siglas como "MDR" generaban capturas erróneas en dominios completamente ajenos al estudio.
- **Procesamiento sin Clasificación.** Consistía en enviar todos los documentos al resto del *pipeline*, garantizando cobertura total. Sin embargo, generaba una sobrecarga computacional y un uso ineficiente de los servicios externos, reduciendo la eficiencia global del sistema.
- **Clasificación Directa con LLM.** La utilización de un LLM sin evidencia contextual mostró alta sensibilidad al diseño de *prompts* y variabilidad en las decisiones de casos límite. La ausencia de documentos de referencia impedía sustentar las decisiones, afectando la consistencia.

### 4.3.3. Arquitectura híbrida adoptada

Los problemas detectados en los enfoques anteriores motivaron la adopción de una arquitectura híbrida que combina filtrado léxico de bajo costo, recuperación de contexto semántico y clasificación final apoyada en evidencia. Este diseño permite ajustar parámetros (umbrales, $\alpha$) sin reentrenar modelos, mantiene costos acotados y se beneficia de un aprendizaje continuo: cada nueva noticia relevante enriquece automáticamente la base vectorial.

**Ventajas de la validación *fuzzy* inicial**

- **Filtrado eficiente**: elimina rápidamente documentos obviamente irrelevantes.
- **Bajo costo computacional**: se ejecuta localmente sin consumo de APIs externas.
- **Buena precisión**: captura variaciones de escritura de organizaciones y siglas del dominio rural uruguayo.

**Ventajas de la búsqueda híbrida posterior**

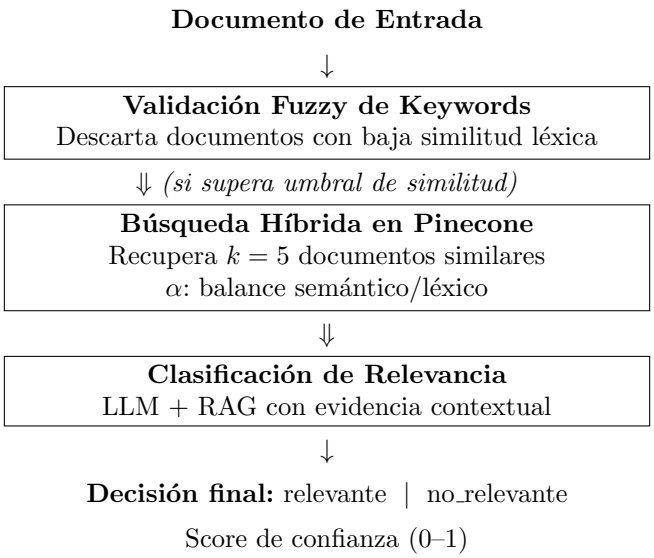
- **Búsqueda semántica con vectores densos**: identifica relevancia aunque el texto use sinónimos o paráfrasis (ej. "protesta" y "manifestación").
- **Búsqueda léxica con vectores dispersos**: garantiza coincidencias exactas en nombres propios y terminología técnica.
- **Complementariedad**: combina precisión léxica y cobertura semántica, superando las debilidades individuales.
- **Base pre-clasificada**: utiliza un corpus de noticias relevantes que provee ejemplos y patrones validados.
- **Contexto enriquecido**: los documentos similares sirven de evidencia para la decisión final.
- **Aprendizaje continuo**: cada nueva noticia relevante se incorpora a la base vectorial, fortaleciendo el sistema con el tiempo.
- **Mantenibilidad**: permite ajustar umbrales y pesos sin reentrenamiento complejo.

**Ventajas del enfoque RAG en la clasificación final**

- **Análisis contextualizado**: combina el contenido del documento con la evidencia recuperada.
- **Decisiones basadas en evidencia**: utiliza patrones reales de la base de conocimiento para fundamentar la clasificación.
- **Manejo de ambigüedad**: los *scores* de similitud ayudan a resolver casos límite.
- **Transparencia**: registra puntajes de similitud y documentos de referencia usados en la decisión.
- **Escalabilidad**: permite incorporar nuevos patrones de relevancia sin actualizar reglas manuales.

### 4.3.4. Implementación final

El flujo de procesamiento se resume en la siguiente secuencia:




Esta implementación combina lo mejor de cada enfoque: un filtro léxico rápido que evita procesar ruido, una búsqueda híbrida que agrega contexto semántico y una clasificación final basada en RAG que fundamenta cada decisión en evidencia empírica, logrando un equilibrio entre eficiencia, precisión y escalabilidad.

### 4.3.5. Tecnologías clave

La solución integra herramientas seleccionadas en función de su robustez y facilidad de mantenimiento. FuzzyWuzzy[3] proporciona algoritmos de coincidencia aproximada con soporte completo de Unicode y umbrales configurables, ideales para un filtrado temprano. Pinecone[4], por su parte, ofrece una API sencilla (*upsert*, filtros, *top-k*) y alto rendimiento para gestionar el índice vectorial sin requerir infraestructura propia. Esta combinación permite un flujo escalable y transparente, donde cada decisión queda respaldada por puntajes de similitud y evidencia documental.

## 4.4. Modulo - Extracción

La etapa de extracción transforma los documentos clasificados como relevantes en salidas estructuradas por dimensiones del dominio: geográfica, acciones, movimientos, institucional y ODS. En el flujo lógico presentado en la Figura 4.2, estas tareas aparecen como "Extractor Geográfico", "Extractor de Acciones", "Extractor de Movimientos", "Extractor de Instituciones" y "Extractor de ODS".

[3]Documentación oficial de la librería: https://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/

[4]Sitio y documentación oficial de Pinecone: https://www.pinecone.io y https://docs.pinecone.io

### 4.4.1. Alternativas evaluadas

- **Diseño monolítico (un solo prompt):** una única invocación que intenta completar todo el *schema* Pydantic de una vez.
- **Diseño modular (cinco extractores):** un extractor por dimensión del dominio con validaciones específicas de Pydantic por etapa.

**Motivación y límites del enfoque monolítico.** La opción de un único prompt obligaba a reunir en una sola instrucción los criterios y validaciones de las cinco dimensiones. En la práctica, ese prompt se volvía demasiado extenso (instrucciones, ejemplos y definiciones por dimensión), presionando el presupuesto de contexto y generando interferencias entre campos; el efecto fue más errores de parseo y menor trazabilidad al depurar. Además, cualquier cambio en una dimensión implicaba reescribir y recalibrar todo el bloque. Dada la complejidad específica de cada dominio, se optó por modularizar. Más adelante se muestran ejemplos de prompts por extractor.

### 4.4.2. Alternativas tecnológicas para el diseño modular

- **HuggingFace + Pydantic + LangChain.** Modelos abiertos (Llama, Mistral, etc.) orquestados con cadenas; Pydantic para salidas tipadas.
- **OpenAI + Pydantic + LangGraph.** Modelos de OpenAI; grafo explícito para el flujo; Pydantic para salidas tipadas.
- **OpenAI + Pydantic + LangGraph + FastMCP.** Igual que lo anterior, sumando herramientas del dominio expuestas vía MCP.

### 4.4.3. Implementación final

**Por qué no HuggingFace + LangChain.** En el corpus (español uruguayo y dominio específico) observamos menor adherencia al formato y mayor variabilidad entre ejecuciones, requiriendo más reintentos para obtener JSON válido por dimensión, lo cual llevaba a un mayor costo operativo. Además, la operación (selección/hosting/actualización de modelos y tooling) elevaba la complejidad sin un beneficio claro de costo frente a la alternativa elegida.

**Por qué LangGraph frente a LangChain.**

- **Flujo visible y condicionable:** definición explícita de nodos y condiciones; rutas y cortes tempranos quedan claras.
- **Estado y reanudación por paso:** *checkpointing* y reintentos a nivel de nodo; reintento del extractor fallido sin repetir todo.
- **Trazabilidad y pruebas:** artefactos por paso (entrada/salida, versión de *schema*, hash de prompt) y pruebas de subrutas.
- **Menos "magia" de agente:** herramientas del dominio se inyectan de forma controlada por nodo (sin decisiones implícitas).

**Por qué OpenAI + LangGraph + FastMCP.**

- **Calidad y estabilidad de salida (OpenAI):** mejor adherencia a instrucciones y menor tasa de JSON inválido → menos reparaciones.
- **Validaciones y enriquecimiento fuera del prompt (FastMCP):** normalización geográfica, reglas de negocio y catálogos, sin "inflar" el prompt.
- **Operación y mantenibilidad:** cambios en una dimensión no afectan al resto; *schemas*/prompts versionados por extractor.
- **Encaje operativo con el grafo definido:** el flujo de LangGraph ya cubre el control fino; el combo solo lo aprovecha (sin repetir lógica).

### 4.4.4. Herramientas empleadas: OpenAI + Pydantic + FastMCP

**LLMs (OpenAI).** Se emplean modelos de lenguaje para mapear texto a estructuras tipadas por dimensión. El *prompting* delimita estrictamente tareas (una dimensión del dominio por vez) y el tamaño de contexto se ajusta a cada extractor.

**Pydantic (salidas tipadas).** Cada dimensión define un *schema* con validaciones (tipos, enums, rangos, formatos). Si la respuesta no valida, se registra el error y se reintenta sólo ese extractor.

**FastMCP (herramientas del dominio).** Se exponen funciones externas para enriquecimiento y verificación, p. ej.: `get_localidad_info(nombre)` (mapear localidad→departamento/región), `get_departamentos_from_region` (consistencia inversa), `validate_action_combination` (reglas de negocio).

### 4.4.5. Flujo operativo y relación con el pipeline

El grafo ejecuta: `classify_relevant_news` → (si relevante) `extract_geographic` → `extract_actions` → `extract_movements` → `extract_institutional` → `extract_ods`. Cuando no hay dependencias lógicas, algunos extractores pueden ejecutarse en paralelo. El contexto recuperado en Clasificación (RAG) se reutiliza en los extractores que se benefician de ese soporte (p. ej., `actions`, `movements`). Figura 4.2.

### 4.4.6. Prompts y ejemplos por extractor

En los listados siguientes, el marcador `{rag_section}` se sustituye por el contexto RAG (resultado similares y metadatos), `{text}` por el documento canónico a procesar. Las herramientas listadas se exponen al agente vía *FastMCP* (protocolo MCP).

**Nota sobre los ejemplos:** A modo ilustrativo, en cada prompt se muestra bajo la sección "Noticia" un fragmento del contenido que corresponde a {`text`}. Los fragmentos mostrados corresponden a porciones específicas de la noticia sobre la solicitud de las gremiales agropecuarias (véase Anexo 7.3).

### Extractor Geográfico

```
"""
Eres un analista experto en geografia territorial de Uruguay y movimientos
    sociales.
Tu tarea es extraer una unica referencia cartografica del siguiente texto,
    estructurada
como objeto JSON valido, cumpliendo las reglas del modelo "
    ReferenciaCartografica".

-- Herramientas disponibles --
- get_localidad_info(nombre: str) -> devuelve region y departamento exactos
    de una localidad
- search_localidad(nombre: str) -> busca coincidencias parciales por nombre
    de localidad
- get_region_by_departamento(departamento: str) -> region correspondiente a
    un departamento
- get_departamento_by_region(region: str) -> departamentos de una region
- validate_geographic_combination(region, departamento, localidad) -> valida
    coherencia

-- Instrucciones paso a paso --
1) Analiza el texto y determina si se menciona una:
   a) Localidad especifica
   b) Departamento claro
   c) Region generica
   d) Referencia nacional ("Uruguay", "todo el pais", multiples departamentos
     de distintas regiones)

2) Segun lo identificado:
- Si hay localidad exacta:
  -> Invoca get_localidad_info(nombre) y usa region, departamento y localidad
     retornados.
- Si hay departamento pero no localidad:
  -> Usa get_region_by_departamento(departamento) para obtener la region y
    deja localidad: "NO ESPECIFICA".
- Si hay solo region:
  -> Usa "departamento": "NO ESPECIFICA" y localidad: null.
- Si hay varios departamentos de distintas regiones:
  -> Responde con "region": "NACIONAL" y los otros campos como null.
- Si no se puede inferir nada:
  -> Usa "region": "NO ESPECIFICA"; los otros campos como null.

3) Valida siempre la combinacion final con validate_geographic_combination.

-- Estructura esperada --
{
  "region": "<Enum Region>",
  "departamento": "<Enum Department | null>",
  "localidad": "<string | null>"
}

Reglas:
- "region" nunca puede ser null (debe ser un valor del Enum Region).
- "departamento": Enum, "NO ESPECIFICA" o null.
- "localidad": string, "NO ESPECIFICA" o null.
- Si una localidad aparece solo en el nombre de una organizacion, no la tomes
     como referencia geografica.

-- Ejemplos validos --
```

```
{"region":"NO ESPECIFICA","departamento":null,"localidad":null}
{"region":"NACIONAL","departamento":null,"localidad":null}
{"region":"SUR","departamento":"NO ESPECIFICA","localidad":null}
{"region":"NORESTE","departamento":"Cerro Largo","localidad":"NO ESPECIFICA"}
{"region":"NOROESTE","departamento":"Artigas","localidad":"Bella Union"}

{rag_section}

-- Noticia --
Debido a los efectos del deficit hidrico en todo el territorio nacional [...]
si bien en anos anteriores se dispusieron beneficios para determinados
    departamentos,
la situacion de sequia alcanza hoy a todo el pais; por ello, solicitan que la
     medida se extienda a todo el territorio. """
```

**Ejemplo de salida (Pydantic).**

```
{
  "region": "NACIONAL",
  "departamento": null,
  "localidad": null
}
```

En este caso, las referencias a "todo el territorio nacional" y "todo el país" indican un alcance geográfico nacional, resultando en la clasificación `NACIONAL` con departamento y localidad marcados como `null`.

**Extractor de Acciones**

```
"""
Analiza la siguiente noticia y extrae la informacion clave sobre la accion
    principal.

Eres un analista experto en las acciones de los movimientos sociales rurales
    de Uruguay.
Debes llenar por completo el modelo Pydantic "Action" con:
- objetivo (ActionObjective)
- matriz   (ActionMatrix)
- derivada (DerivedActionType | null)

-- Herramientas disponibles --
- get_action_matrix_context  -> Dict[str, {title:str, criteria:list[str]}]
- get_derived_action_context -> Dict[str, {title:str, criteria:list[str],
    matrix:str}]
- validate_action_combination(matriz:str, derivada:str)

-- Instrucciones paso a paso --
1) Invoca una sola vez get_action_matrix_context y una sola vez
    get_derived_action_context.
2) Determina:
   a) objetivo: elige EXACTO entre
      "Propositiva","Resistencia","Defensiva (reactiva)","Reivindicativa",
      "Emancipatorios (superadores)","Reprimir","Negociar","Cooptar"
   b) matriz: elige EXACTO entre
      "No especifica","No corresponde","Manifestacion/Protesta","Vinculo con
    el Estado",
```

```
      "Entramados Socio-Culturales","Comercializacion","Produccion","Consumo
    ",
      "Acciones con otras organizaciones o movimientos"
   c) derivada: elige EXACTO (o null) entre
      "Marcha","Paro","Movilizacion","Pronunciamientos","Pedido de acceso de
    informacion",
      "Proyectos","Espacios de participacion","Integracion de organismos
    publicos",
      "Resolucion","Eventos","Festividades","Acciones comunitarias para
    acceso a servicios",
      "Circuitos Cortos de Comercializacion","Compras publicas","Venta
    conjunta",
      "Compra conjunta","Ferias agroecologicas","Canastas","Estrategias de
    apoyo a la produccion",
      "Campos colectivos","Maquinaria colectiva","Redes productor-consumidor
    ",
      "Certificacion participativa","Visitas de predios"
3) Llama a validate_action_combination(matriz, derivada); si no es valida,
    usa derivada = null.
4) Devuelve EXCLUSIVAMENTE un JSON valido:
{
  "objetivo": "<ActionObjective>",
  "matriz": "<ActionMatrix>",
  "derivada": "<DerivedActionType | null>"
}

Reglas estrictas:
- Usa exactamente los valores de los Enums.
- Si derivada no es compatible con matriz, debe ser null.
- No agregues texto fuera del bloque JSON.

{rag_section}

-- Noticia --
solicitaron al Poder Ejecutivo una reduccion de la alicuota de la
    Contribucion Inmobiliaria Rural [...] El pedido fue dirigido al presidente
     de la Republica, Luis Lacalle Pou; a la ministra de Economia y Finanzas (
    MEF), Azucena Arbeleche; y al ministro de Ganaderia, Agricultura y Pesca (
    MGAP) [...] las gremiales aguardan una definicion a corto plazo. """
```

**Ejemplo de salida (Pydantic).**

```
{
  "objetivo": "Reivindicativa",
  "matriz": "Vinculo con el Estado",
  "derivada": null
}
```

En este caso, la acción de "solicitar al Poder Ejecutivo" una reducción impositiva constituye una acción `Reivindicativa` dirigida al Estado, clasificándose como `Vínculo con el Estado` sin derivada específica (`null`) al no encajar en ninguna de las subcategorías definidas para este tipo de matriz.

**Extractor de Movimientos**

```
"""
Eres un analista experto en movimientos socioterritoriales rurales de Uruguay
    .
Debes extraer TODO lo necesario para llenar el modelo Pydantic "Movements".

-- Herramientas --
- get_movement_type_context -> Dict[str, {title:str, criteria:list[str]}]

-- Instrucciones --
1) Invoca una sola vez get_movement_type_context.
2) Determina el MovementType con evidencia mas fuerte (si no, "No Corresponde
    ").
3) Identifica hasta dos organizaciones/colectivos explicitos (orden de
    relevancia).
   - Usa exactamente los nombres del Enum MovementName.
   - Si no esta en la lista: "nombre":"OTROS" y coloca el real en "
    otros_movimientos".
   - Si no hay organizaciones: "nombre":"S_D".
4) Devuelve EXCLUSIVAMENTE un JSON valido:
{
  "tipo": "<MovementType>",
  "movimientos": [
    {"nombre":"<MovementName|OTROS|S_D>", "otros_movimientos":"<string|null
    >"}
  ]
}

Reglas estrictas:
- "tipo" debe ser uno de:
  "Movimientos de la agricultura familiar-campesina","Movimientos
    Agroecologicos",
  "Movimientos Ruralista-Agronegocio","Movimientos Sindical Rural y
    Agroindustrial",
  "Movimientos Mujeres Rurales","Movimientos Ambientalistas",
  "Movimientos Juventudes Rurales","Movimiento Cooperativo",
  "Movimiento de Pueblos Originarios","Movimiento Afro",
  "Movimiento de Consumo Alimentario","No Corresponde"
- Si "nombre" != "OTROS" entonces "otros_movimientos" = null.

{rag_section}

-- Noticia --
Dos de las principales gremiales del sector agropecuario, la Asociacion Rural
del Uruguay (ARU) y la Federacion Rural (FR) --integrantes de Campo Unido--
solicitaron al Poder Ejecutivo [...] La carta fue firmada por Gonzalo Valdes
Requena, presidente de la ARU, y por Martin Uria Shaw, presidente de la FR.
    """
```

**Ejemplo de salida (Pydantic).**

```
{
  "tipo": "Movimientos Ruralista-Agronegocio",
  "movimientos": [
    {"nombre": "OTROS", "otros_movimientos": "OTROS"},
    {"nombre": "Asociacion Rural del Uruguay", "otros_movimientos": "
    Federacion Rural"}
  ]
```

```
}
```

En este caso, la mención explícita de “Asociación Rural del Uruguay (ARU)” y “Federación Rural (FR)” como “principales gremiales del sector agropecuario” las clasifica como `Movimientos Ruralista-Agronegocio`. El primer movimiento se categoriza genéricamente como `OTROS`, mientras que ARU aparece explícitamente con `Federación Rural` en el campo `otros_movimientos`.

### Extractor Institucional

```
"""
Eres un analista experto en instituciones y politicas publicas de Uruguay.
Debes extraer exactamente UNA institucion mencionada y devolverla como JSON
segun el modelo Pydantic "Institution".

-- Herramientas disponibles --
- get_state_action_context -> Dict[str, {title:str, description:str, criteria
    :list[str]}]
  (cada clave corresponde a un Enum StateActionType)

-- Instrucciones paso a paso --
1) Invoca una sola vez get_state_action_context.
2) Determina:
   a) tipo_de_institucion: usa EXACTO el Enum InstitutionalType (lista en el
    sistema).
   b) nombre_de_institucion: nombre completo si aparece; si no, null.
   c) tipo_de_accion_de_estado: compara con criterios de StateActionType (o "
    No corresponde").

Devuelve EXCLUSIVAMENTE un JSON:
{
  "tipo_de_institucion": "<InstitutionalType>",
  "otro_tipo": "<string | null>",
  "nombre_de_institucion": "<string | null>",
  "tipo_de_accion_de_estado": "<StateActionType>"
}

Reglas estrictas:
- Si "nombre_de_institucion" esta presente, "tipo_de_institucion" no puede
    ser "No corresponde".
- Si "tipo_de_institucion" = "Otros" entonces "otro_tipo" es obligatorio; en
    caso contrario, null.
- Si no identificas ninguna institucion, devuelve {}.
- No agregues texto fuera del bloque JSON.

{rag_section}

-- Noticia --
El pedido fue dirigido al presidente de la Republica, Luis Lacalle Pou;
a la ministra de Economia y Finanzas (MEF), Azucena Arbeleche; y al ministro
de Ganaderia, Agricultura y Pesca (MGAP), Fernando Mattos [...] ambos
ministerios acusaron recibo del planteo; por el momento no hubo respuesta.
"""
```

### Ejemplo de salida (Pydantic).

```
{
  "tipo_de_institucion": "Presidencia de la Republica",
  "otro_tipo": null,
  "nombre_de_institucion": "Presidencia de la Republica",
  "tipo_de_accion_de_estado": "No corresponde"
}
```

En este caso, la mención del "presidente de la República, Luis Lacalle Pou" como destinatario del pedido permite identificar la `Presidencia de la República` como institución principal. El `tipo_de_accion_de_estado` se clasifica como `No corresponde` ya que la institución aparece como receptora de una solicitud, no ejecutando una acción activa.

### Extractor de ODS

```
"""
Analiza la noticia y extrae la informacion sobre Objetivos de Desarrollo
    Sostenible (ODS).
Eres un analista experto en ODS (ONU) y en acciones de movimientos sociales
    de Uruguay.

-- Herramienta disponible --
- get_ods_context -> Dict[str, {title:str, description:str, criteria:list[str
    ]}]

-- Instrucciones paso a paso --
1) Invoca una sola vez get_ods_context.
2) Busca coincidencias (terminos, sinonimos, temas) entre el texto y los
    criterios.
3) Elige el ODS con evidencia mas solida (si no hay, "NO_ESPECIFICA").
4) Devuelve EXCLUSIVAMENTE este JSON:
{"ods": "<ODS1..ODS17 | NO_ESPECIFICA>"}

Reglas estrictas:
- "ods" debe ser exactamente un nombre de Enum ODS (ODS1...ODS17) o "
    NO_ESPECIFICA".
- No agregues texto fuera del bloque JSON.

{rag_section}

-- Noticia --
Debido a los efectos del deficit hidrico en todo el territorio nacional [...]
considerando que persisten las dificultades generadas por la sequia [...] la
situacion actual de sequia rige en todo el pais. """
```

### Ejemplo de salida (Pydantic).

```
{
    "ods": "ODS13"
}
```

En este caso, las múltiples referencias a "sequía" y "déficit hídrico" como problemática que afecta "todo el territorio nacional" se alinean directamente con el `ODS13: Adoptar medidas urgentes para combatir el cambio climático y sus efectos`, específicamente con el criterio de "cuestiones de cambio climático, sequía e inundaciones".

## 4.5. Módulo - Orquestación

La orquestación asegura que el procesamiento de noticias se ejecute de forma periódica, distribuida y tolerante a fallos. Coordina el encolado de trabajos, aplica mecanismos de reintento e idempotencia y organiza la ejecución en colas especializadas. En esta arquitectura, *Celery* actúa como motor de ejecución distribuida y *Celery Beat* como planificador periódico de trabajos.

### 4.5.1. Alternativas evaluadas

A continuación se sintetizan las opciones consideradas para estructurar la ejecución.

**(a) Cola única con *workers* monolíticos.** Una única cola y una tarea "de extremo a extremo" por artículo.

- **Ventajas:** simplicidad y despliegue rápido.
- **Limitaciones:** baja observabilidad por etapa, difícil imponer *rate limits* por proveedor y reintentar sólo el tramo fallido; acoplamiento entre coordinación y procesamiento.

**(b) Colas diferenciadas y *scheduling* explícito.** Separación del flujo en colas distintas (coordinación/ingesta y procesamiento) con programación explícita vía *Beat*.

- **Ventajas:** control fino de concurrencia, trazabilidad por etapa, reintentos segmentados e independencia entre coordinación y procesamiento.
- **Limitaciones:** mayor complejidad operativa y necesidad de observabilidad más detallada.

### 4.5.2. Diseño adoptado

Se adopta la opción **(b) colas diferenciadas y *scheduling* explícito**. En lo que sigue se detallan las decisiones de diseño resultantes.

- **Separación de responsabilidades.** Una cola de coordinación ejecuta la ingesta periódica de alertas; una cola de procesamiento ejecuta, por artículo, el *pipeline*.
- **Concurrencia y *rate limits*.** Concurrencia parametrizable por cola/*worker*, evitando picos y respetando cuotas de terceros.
- **Reintentos e idempotencia.** Políticas de *backoff* y deduplicación por identificador de artículo para reducir reprocesamientos ante fallas transitorias.
- **Escalabilidad y mantenibilidad.** Incorporación de nuevas colas o etapas sin modificar el diseño general; desacople que facilita pruebas y despliegues incrementales.

### 4.5.3. Tareas principales

El sistema implementa dos tareas complementarias que materializan el ciclo completo de ingesta y procesamiento.

**(a) Tarea de coordinación e ingesta (`gmail_coordinator_task`).** Se ejecuta de manera periódica (disparada por Celery Beat en un intervalo fijo). Su función es consultar la fuente de alertas (p. ej., Google Alerts), transformar los hallazgos en artículos candidatos y encolar una tarea de procesamiento por cada artículo. Para evitar sobrecargas, la programación de las tareas posteriores se escalona mediante pequeños retrasos (*countdown*) entre ejecuciones.

**(b) Tarea de procesamiento individual (`extraction_task`).** Ejecuta directamente el **pipeline de ejecución** definido por los módulos de *Ingesta*, *Clasificación* y *Extracción*. Todos los resultados se almacenan en MongoDB; los casos exitosos (`TaskSuccessResult`) también se envían a la JotForm API para su posterior análisis y reporte.

### 4.5.4. Flujo extremo a extremo

Para claridad, se describe el recorrido típico de un documento a través del sistema.

1. **Programación:** Celery Beat dispara `gmail_coordinator_task` a intervalos regulares.
2. **Ingesta y encolado:** la tarea coordinadora recupera nuevas alertas y encola, con ejecución escalonada, una `extraction_task` por artículo.
3. **Procesamiento:** cada `extraction_task` es tomada por un *worker* y ejecuta el pipeline (Ingesta → Clasificación → Extracción), aplicando reintentos ante fallos transitorios.
4. **Persistencia y confirmación:** los resultados se almacenan y el estado de la tarea queda disponible para monitoreo y análisis.

## 4.6. Pipeline de procesamiento

El *pipeline* de procesamiento presentado en la Figura 4.2 describe la trayectoria de una noticia desde la ingesta hasta su persistencia en bases de datos y servicios externos. La arquitectura final organiza el flujo en cuatro etapas principales:

1. **Ingesta**: obtiene los documentos desde el servicio Gmail mediante su API y ejecuta el módulo de *scraping* para la extracción y normalización del contenido web de las noticias.
2. **Clasificación**: implementa un filtro de tres niveles secuenciales para determinar la relevancia del documento:
   - **Validación de *keyword***: aplica coincidencia difusa (*fuzzy matching*) con un umbral del 75
   - **Búsqueda híbrida en Pinecone**: combina búsqueda semántica (mediante *embeddings* densos) y léxica (mediante *sparse embeddings*) consultando la base vectorial externa.
   - **Clasificador de relevancia**, potenciado por modelos de lenguaje.

   Solo los documentos que superan los tres filtros continúan al proceso de extracción.

3. **Extracción con MCPs**: constituye el núcleo semántico del sistema. Incluye:
    - Un módulo de Extracción con RAG, recupera documentos similares desde Pinecone para enriquecer el contexto de todos los extractores.
    - Extractores por dimensiones específicas que estructuran la información en el dominio del OCAU: Acciones, Movimientos Sociales, Instituciones, ODS y Geografía.
4. **Tipos de resultado**: el *pipeline* concluye con la categorización del resultado en diferentes categorias:
    - ***TaskSuccessResult***: extracción completada exitosamente con todas las dimensiones procesadas.
    - ***TaskValidationFailedResult***: documento descartado por no superar los filtros de relevancia.
    - ***TaskScrapingFailedResult***: fallo en la obtención del contenido web durante la ingesta.
    - ***TaskRagFailedResult***: error durante el procesamiento RAG o en los extractores MCP.

    Todos los resultados se almacenan en MongoDB, mientras que los casos exitosos también son enviados a JotForm API.

La **Pinecone DB** aparece como servicio externo, que nutre tanto la clasificación (para evaluar similitud con noticias previas) como en la extracción (para enriquecer el contexto mediante RAG).

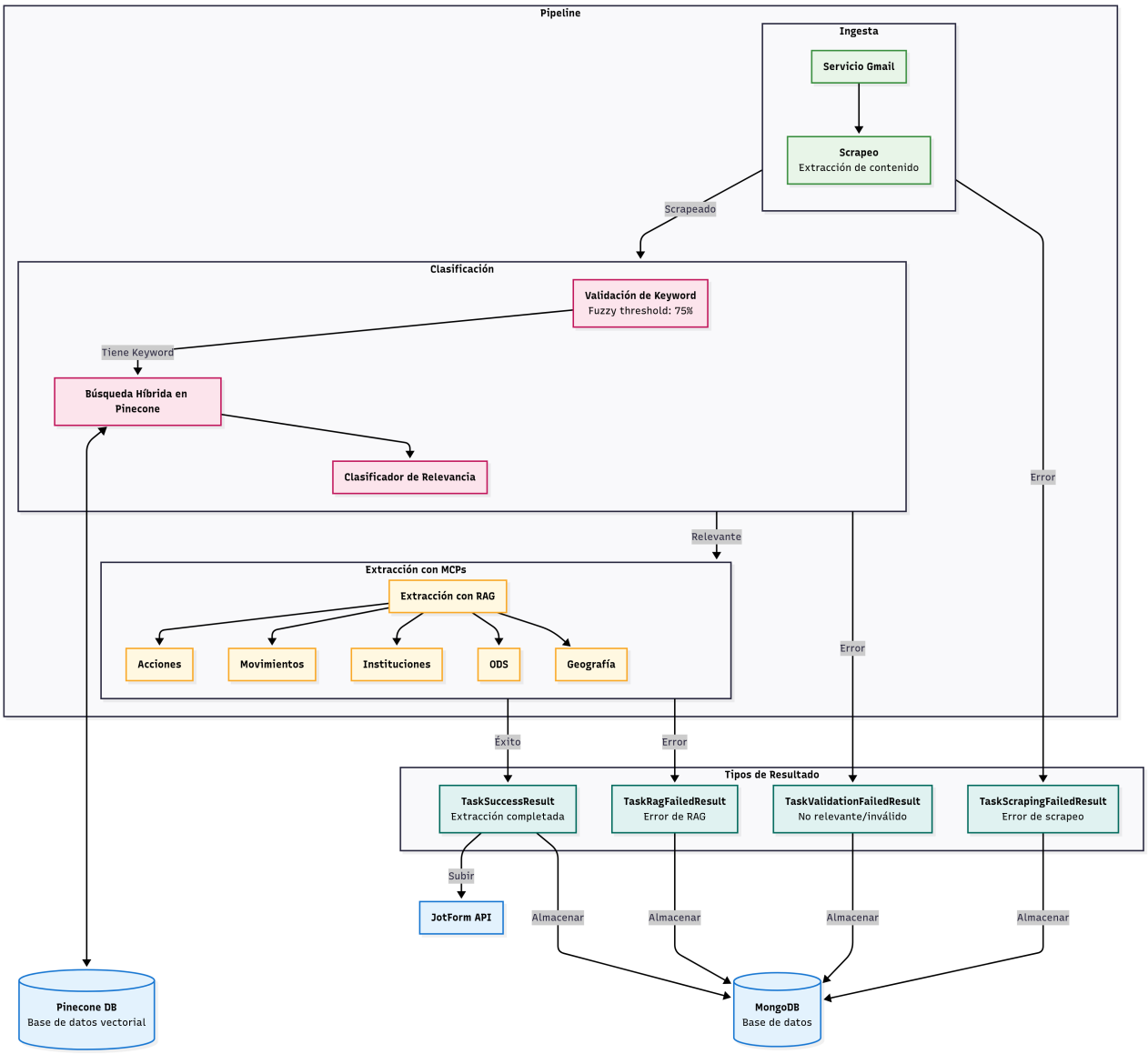



Figura 4.2: Pipeline de procesamiento

## 4.7. API Backend

La API Backend constituye la capa de presentación del sistema, proporcionando una interfaz REST para la interacción y la exposición de los resultados del procesamiento con el frontend. Este módulo se desarrolló en Python utilizando FastAPI[5] como *framework web*,

[5] https://fastapi.tiangolo.com/

elegido por su facilidad de uso, su buen soporte para operaciones asíncronas, su integración directa con Pydantic para validación y tipado de datos. Estas características lo hacen especialmente adecuado para construir APIs modernas y eficientes. Para exponer el servicio se utilizó Uvicorn[6] como servidor ASGI (*Asynchronous Server Gateway Interface*), una opción liviana y rápida que facilita la concurrencia y mejora el rendimiento en aplicaciones web asíncronas.

### 4.7.1. Función

La API Backend cumple tres funciones principales dentro de la arquitectura del sistema:

- **Gestión de Autenticación y Autorización**: Implementa un sistema de autenticación que garantiza el acceso controlado a los recursos del sistema, permitiendo el registro y *login* de usuarios.
- **Exposición de Resultados**: Proporciona endpoints REST para consultar los resultados del procesamiento de noticias, incluyendo funcionalidades de paginación, filtrado y estadísticas. Los datos se estructuran en respuestas JSON estandarizadas con metadatos de paginación.
- **Control Operacional**: Ofrece endpoints para monitorear el estado del sistema y controlar la ejecución del pipeline de procesamiento, permitiendo habilitar/deshabilitar el *beat* de Celery de forma dinámica mediante *feature flags*.

#### Implementación Final

La implementación de la API Backend se estructura en una arquitectura modular con tres componentes principales:

- **Aplicación Principal**: Define la instancia FastAPI con configuración de CORS para acceso controlado, y gestión del ciclo de vida que inicializa la conexión a MongoDB durante el *startup* del sistema.
- **Sistema de Rutas**: La API organiza su funcionalidad en cuatro *routers* bajo el prefijo `/api/v1`:
  - **auth**: Registro de usuarios, login mediante OAuth2, consulta de información del usuario autenticado y *logout*.
  - **task**: Consulta y gestión de resultados del procesamiento PLN con soporte para paginación, estadísticas y operaciones CRUD.
  - **user**: Gestión de perfiles de usuario y funcionalidades administrativas.
  - **beat**: Control dinámico del pipeline de procesamiento mediante *feature flags* en Redis.
- **Seguridad**: Implementa un sistema de autenticación basado en JWT (*JSON Web Tokens*) que utiliza OAuth2 con flujo de contraseñas para el login de usuarios. Las credenciales se almacenan hasheadas con bcrypt en MongoDB. Los endpoints protegidos validan automáticamente tokens JWT en cada petición, extrayendo la información del usuario para autorización posterior.

[6] https://uvicorn.dev/

## 4.8. Frontend

El *frontend* constituye la interfaz visual del sistema, diseñada para proporcionar a los usuarios una experiencia intuitiva en el monitoreo de resultados de extracción, visualización de estadísticas generales, y gestión del estado operacional de la herramienta.

Este módulo consume los servicios que expone el *backend*, presentando la información de manera clara y accesible para facilitar la interpretación de los resultados del *pipeline* de extracción.

### 4.8.1. Funcionalidades Principales

La aplicación web proporciona las siguientes capacidades:

- **Dashboard de Monitoreo**: Ofrece una vista general del estado del sistema con estadísticas en tiempo real sobre el procesamiento de noticias, incluyendo métricas de éxito, errores y validaciones fallidas. Además, permite gestionar el encendido y apagado de la herramienta de extracción mediante un control centralizado.

- **Gestión de Noticias Procesadas**: Proporciona una tabla interactiva con todas las noticias que han sido procesadas por el sistema. Incluye capacidades de filtrado por estado de procesamiento (éxito, error, validación fallida), paginación, y visualización detallada de cada extracción.

- **Sistema de Autenticación**: Implementa un sistema de gestión de usuarios con *login* y administración de sesiones, garantizando que solo usuarios autorizados puedan acceder a las funcionalidades del sistema.

- **Visualización Detallada**: Cada noticia procesada puede ser examinada en detalle, mostrando información completa sobre la extracción realizada, puntuación de relevancia, coincidencias encontradas y metadatos asociados.

### 4.8.2. Consideraciones Técnicas

La interfaz fue desarrollada utilizando React[7] versión 19 con TypeScript[8], proporcionando un entorno de desarrollo robusto con tipado estático que reduce errores y mejora la experiencia de desarrollo. Se seleccionó *Vite*[9] como *bundler* por su velocidad de compilación y recarga rápida durante el desarrollo, además de generar *builds* optimizadas para producción.

Para el enrutamiento se optó por TanStack Router[10], una solución que proporciona enrutamiento basado en archivos y generación automática de rutas tipadas. Esta elección garantiza seguridad de tipos *end-to-end* en toda la navegación de la aplicación.

La comunicación con el *backend* se realiza mediante *Axios*, una librería HTTP que facilita la configuración de interceptores para manejo de *tokens* de autenticación, manejo de errores centralizado, y transformación de respuestas.

[7] https://react.dev/
[8] https://www.typescriptlang.org/
[9] https://vite.dev/
[10] https://tanstack.com/router

Para el manejo de tablas se utilizó *TanStack Table*[11]. Por su parte, *TanStack Query*[12] gestiona el estado asíncrono de las llamadas al *backend*.

El sistema de diseño se construyó sobre Tailwind CSS[13], un *framework* de CSS basado en clases utilitarias que permite construir interfaces directamente en el marcado HTML sin necesidad de escribir CSS personalizado.

La gestión del estado global se implementó con Zustand[14], una librería ligera que ofrece una API simple para manejar estado compartido en la aplicación, especialmente útil para la gestión de sesiones de usuario y configuraciones globales.

La aplicación se despliega como una *Single Page Application* (SPA) que se comunica con el *backend* a través de una API REST, manteniendo una separación clara entre *frontend* y *backend*.

### 4.8.3. Estructura del Proyecto

La arquitectura del *frontend* se organizó siguiendo principios de modularidad y separación de responsabilidades, facilitando el mantenimiento y escalabilidad del código. A continuación se describe la función de cada carpeta principal:

- **components**: Almacena componentes reutilizables organizados por funcionalidad que son usados en diversas páginas. Estos componentes modulares y autónomos optimizan el desarrollo y aseguran consistencia visual y funcional en la aplicación.
- **hooks**: Contiene *custom hooks* que encapsulan lógica reutilizable. Los *hooks* de API gestionan las interacciones con el *backend* utilizando *Tanstack Query* para manejo eficiente de estado asíncrono y caché.
- **lib**: Incluye configuraciones y utilidades. Define servicios de API con *Axios* para comunicación con el *backend*.
- **routes**: Aprovecha el sistema de enrutamiento basado en archivos de *TanStack Router*. Cada archivo representa una ruta de la aplicación, con rutas protegidas organizadas para garantizar que solo usuarios autenticados accedan a ellas.
- **stores**: Gestiona el estado global de la aplicación mediante *Zustand*. Este maneja la autenticación y persistencia de sesión del usuario, proporcionando un punto centralizado para el manejo del estado de autenticación.
- **types**: Define las interfaces y tipos *TypeScript* que modelan las entidades del sistema. Estos tipos coinciden con las estructuras de datos del *backend*, asegurando coherencia en la comunicación entre ambos módulos.

#### Gestión del Estado

La aplicación implementa una estrategia híbrida de gestión de estado que combina diferentes soluciones según las necesidades específicas:

[11] https://tanstack.com/table
[12] https://tanstack.com/query
[13] https://tailwindcss.com/
[14] https://zustand.docs.pmnd.rs/

- **Estado Global con Zustand**: Se utiliza para manejar el estado de autenticación del usuario, proporcionando una solución ligera y eficiente que persiste información de sesión mediante *localStorage*.
- **Estado del Servidor con Tanstack Query**: Gestiona toda la comunicación con el *backend*, proporcionando caché automático, sincronización inteligente, y manejo de estados de carga y error. Los *hooks* personalizados encapsulan las *queries* y *mutations*, facilitando el acceso a datos del servidor desde cualquier componente.
- **Estado Local con React Hooks**: Para estado específico de componentes se utilizan *hooks* nativos de *React* como *useState* y *useEffect*.

Esta estrategia permite una clara separación entre diferentes tipos de estado.

### 4.8.4. Secciones de la Aplicación

La interfaz web está compuesta por las siguientes secciones principales:

#### Página de Inicio

Presenta una vista general de la plataforma explicando su propósito y funcionalidades principales. Proporciona un botón de "Iniciar Sesión"que redirige al usuario a la pantalla de autenticación.


Figura 4.3: Página de inicio de la aplicación

#### Inicio de Sesión

Formulario de autenticación que solicita nombre de usuario y contraseña. Permite a los usuarios registrados acceder a las funcionalidades protegidas del sistema. Incluye la opción de volver a la página de inicio.

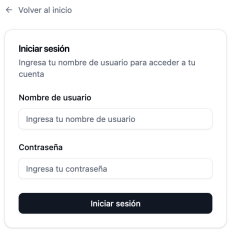



Figura 4.4: Formulario de inicio de sesión

### Dashboard

Panel principal de control que muestra el estado actual del sistema. Presenta tres métricas principales: Total de Noticias Procesadas, Noticias Completadas con porcentaje de éxito, y Noticias Inválidas/Fallidas con porcentaje de error. Incluye un control para iniciar o detener la herramienta de procesamiento de noticias, permitiendo a los usuarios gestionar el estado operacional del sistema en tiempo real.

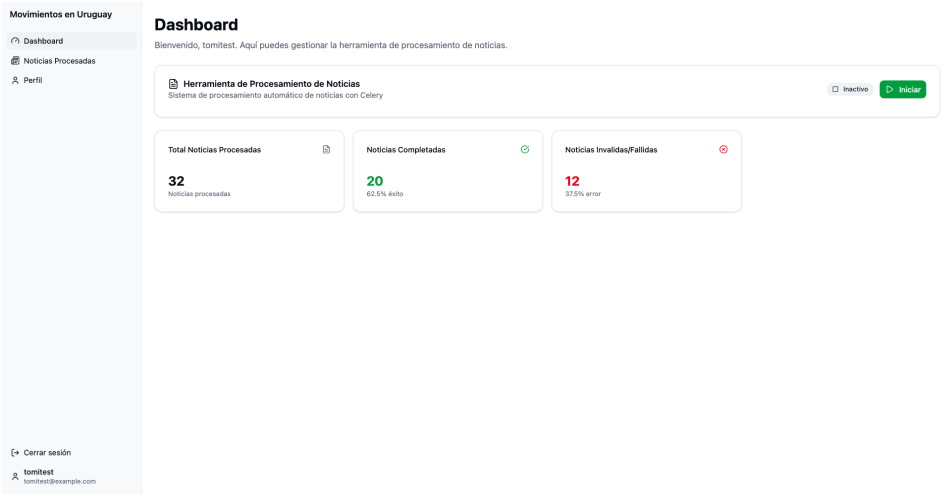



Figura 4.5: Dashboard principal con métricas del sistema

### Noticias Procesadas

Tabla que lista todas las noticias que han sido procesadas por el sistema. Cada fila muestra información clave como el estado de procesamiento (Success, Error, Invalid), título de la noticia, URL, palabra clave asociada, resultado de extracción, relevancia, puntuación, coincidencias encontradas y mensajes de error cuando corresponde. La tabla incluye paginación

para navegar entre múltiples páginas de resultados, y cada noticia cuenta con un botón "Ver detalles"que permite acceder a información más específica.

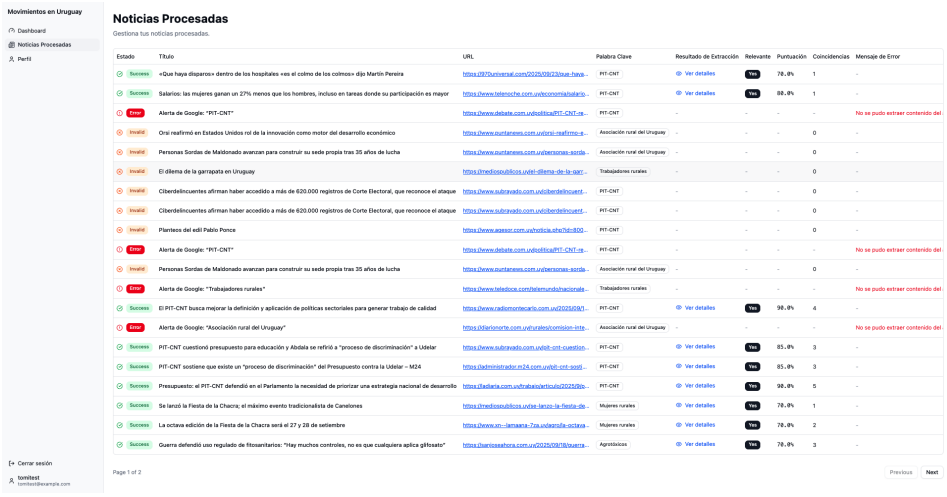



Figura 4.6: Tabla de noticias procesadas con paginación

### Detalle de Noticia

Página de visualización detallada de una noticia específica. Presenta el artículo original con su título, URL y palabra clave asociada. Incluye secciones estructuradas que muestran: Análisis de Relevancia con score y nivel de confianza, Información Geográfica, Acciones y Objetivos, Movimientos Sociales, Objetivos de Desarrollo Sostenible (ODS) relacionados, e Información Institucional sobre las entidades participantes. También muestra métricas de procesamiento como *matches* encontrados, tiempo de procesamiento, secuencia y total de artículos procesados.

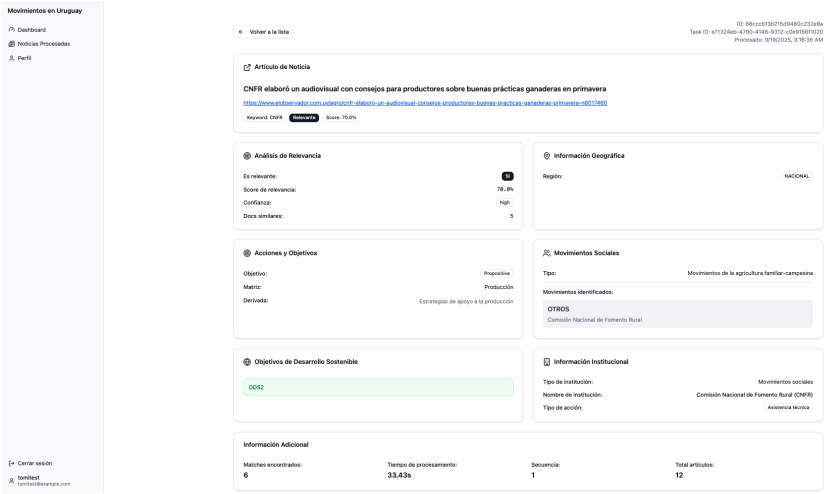



Figura 4.7: Vista detallada de una noticia procesada

# Capítulo 5

# Análisis de resultados

En este capítulo se presentan los resultados obtenidos por el sistema de extracción automática de información a partir de un conjunto de documentos previamente analizados por el OCAU, poniendo a prueba la implementación final del sistema. El objetivo es evaluar el desempeño de la herramienta frente a textos reales, analizando tanto sus aciertos como sus limitaciones.

## 5.1. Ambigüedad en las extracciones

Uno de los aspectos centrales que se observó durante la evaluación es la presencia de ambigüedad en las categorías del dominio. Muchas de las definiciones empleadas por el OCAU no son absolutas, sino que dependen en gran medida de la interpretación del lector. De hecho, al comparar anotaciones realizadas por distintos investigadores sobre un mismo texto, se identificaron discrepancias en la clasificación de ciertas dimensiones, lo cual evidencia que el problema no es exclusivo del sistema automático, sino inherente al propio proceso de análisis humano.

En algunos casos, la ambigüedad surge por la cercanía terminológica entre conceptos, donde las diferencias son sutiles o incluso imperceptibles. Ejemplos típicos son la distinción entre una **marcha** y una **manifestación**, o entre acciones de **resistencia** y de carácter **reivindicativo**. En estos escenarios, resulta posible defender más de una etiqueta como válida según el énfasis interpretativo adoptado.

Asimismo, el sistema de extracción enfrenta limitaciones propias de los modelos de lenguaje. Aun cuando se le proporcionó contexto, herramientas MCP y prompts diseñados con detalle, su conocimiento del dominio es indirecto y depende de las instrucciones y recursos disponibles. Esto genera un doble sesgo: por un lado, el sesgo interpretativo propio del campo de estudio; por otro, el sesgo técnico del modelo al apoyarse en representaciones aproximadas del dominio.

Por estas razones, en la evaluación se adoptó un criterio flexible que reconoce no solo los aciertos exactos, sino también las coincidencias parciales o alternativas plausibles. De esta manera, se busca reflejar la complejidad del problema y valorar la capacidad del sistema de ofrecer lecturas defendibles aun cuando no coincidan palabra por palabra con el estándar de referencia.

## 5.2. Metodología de evaluación

Para llevar a cabo la evaluación, se seleccionaron cinco documentos que fueron previamente analizados, y se evaluó como la herramienta performo en comparación a los investigadores del OCAU.

Cada documento fue procesado por el sistema, y se presentan los resultados en tablas simplificadas, las cuales resumen la información de manera estructurada por dominio.

Además, se incluye una breve **conclusión** en cada caso, donde se discuten los aciertos y limitaciones del resultado, considerando el contexto del texto fuente.

## 5.3. Casos de prueba

A continuación, se presentan los cinco documentos seleccionados junto con sus respectivos resultados de extracción.

### 5.3.1. Documento 1 — Comparación detallada

**Texto fuente**

> El movimiento Un Solo Uruguay (USU), integrado por actores del medio agropecuario, comercial e industrial y con representación en el directorio del Banco de Previsión Social (BPS), acudió a la Comisión de Seguridad Social del Parlamento para opinar sobre la reforma propuesta por el Gobierno de Luis Lacalle Pou y planteó que "no ataca los verdaderos problemas de las jubilaciones y pensiones". Por ese motivo, USU, representado por el productor agropecuario Federico Hoffman, se opone al proyecto.
>
> Según USU, la iniciativa "intenta", mediante la modificación de aspectos jubilatorios como la suba de la edad de retiro y el cálculo del básico jubilatorio, reducir el gasto del BPS. No sería, entonces, una reforma a favor de los trabajadores, sino "un ahorro para el Estado, en demérito de quienes trabajan y generan divisas para el país". Asegura que en la reforma "solo se cuida la parte fiscal" y "el déficit fiscal". Afirma: "Hoy las jubilaciones son miserables y para el futuro serán aún más bajas, porque según quienes proponen esta reforma, si no lo hacen hay riesgo de no cobrar esas prestaciones".
>
> Respecto a las pensiones por viudez, USU señala que no están aseguradas de forma uniforme: hasta los 49 años la prestación sería por 1 año; entre 50 y 54 por 3 años; y a partir de los 55 años en forma permanente.
>
> USU cuestiona además la hipoteca inversa, "creación de esta reforma", destinada a quienes tengan una pasividad baja (personas que cobran hasta 25 000 pesos por mes). Estas personas podrían hipotecar su propiedad ante una entidad financiera autorizada y recibir un monto mensual hasta su fallecimiento; luego, los herederos deberían devolver lo percibido en un plazo de 90 días para conservar el bien, de lo contrario la entidad pasaría a ser propietaria. Para USU, esto evidencia que las futuras jubilaciones serán "muy bajas".
>
> También critica la Agencia Reguladora de la Seguridad Social, "una entidad con directorio cuyo presidente elegirá el presidente de la República de turno y con mucho poder". Estaría por encima del BPS y de los demás sistemas de seguridad social, controlaría la parte financiera y podría intervenir si las finanzas no fueran acordes a las prestaciones. Incluso podría modificar parámetros legales como la edad mínima de jubilación (con piso pero sin techo), lo que, para USU, permitiría que se requiera llegar a 70 años para acumular 30 años de trabajo documentados.

> USU concluye que se opone a la reforma y reclama una reforma integral y de fondo: política fiscal previa, aporte único, jubilación pública única más allá de salarios y aportes, y un seguro privado voluntario para quienes deseen complementar su jubilación.
>
> *Montevideo Portal*[1]

**Geográfica**

| Subclave | Esperado | Extraído |
|---|---|---|
| Región | SUR | NACIONAL |
| Departamento | MONTEVIDEO | – |
| Localidad | NO ESPECIFICA | – |

**Acciones**

| Subclave | Esperado | Extraído |
|---|---|---|
| Objetivo | Resistencia | Reivindicativa |
| Matriz | Vínculo con el Estado | Vínculo con el Estado |
| Derivadas | Espacios de participación | Espacios de participación |

**Movimientos**

| Subclave | Esperado | Extraído |
|---|---|---|
| Tipo | Movimiento Ruralista-Agronegocio (MRA) | Movimientos Ruralista-Agronegocio |
| Nombre | Un Solo Uruguay | OTROS; Un Solo Uruguay |

**Institucional**

| Subclave | Esperado | Extraído |
|---|---|---|
| Tipo | Otras | Movimientos sociales |
| Otro tipo | Comisión de Seguridad Social del Parlamento | – |
| Nombre | – | Un Solo Uruguay |
| Acción de Estado | No Corresponse | No corresponde |

**ODS**

| Subclave | Esperado | Extraído |
|---|---|---|
| Código | ODS8 | ODS10 |
| Descripción | Promover el crecimiento económico inclusivo y sostenible, el empleo y el trabajo decente para todos | Reducir la desigualdad en y entre los países |

**Conclusión**

El modelo captura bien el encuadre: USU comparece ante una comisión parlamentaria y se posiciona contra la reforma; la etiqueta “Nacional” prioriza el alcance político del hecho

[1] https://www.montevideo.com.uy/Noticias/Un-Solo-Uruguay-se-opone-a-la-Reforma-de-Seguridad-Social-es-un-ahorro-para-el-Estado--uc848348

por encima del lugar físico (Montevideo), una lectura aceptable en este contexto. En acciones, "reivindicativa" frente a "resistencia" es una diferencia de rótulo más que de sentido: en ambos casos hay oposición con demanda hacia el Estado; la derivada que propone ("integración de organismos públicos") puede leerse como otra forma de participación institucional (audiencias, comisiones). Reconoce al actor principal (USU), aunque pierde especificidad al tipificarlo genéricamente como "movimientos sociales" en lugar de ruralista-agronegocio. El desliz claro está en lo institucional, donde confunde actor social con la entidad estatal pertinente (debió identificar a la Comisión de Seguridad Social). En ODS, la elección de ODS10 en lugar de ODS8 es discutible pero defendible: el texto habilita una lectura laboral/productiva (trabajo y crecimiento) y otra distributiva (desigualdad y efectos regresivos). En suma, el sistema acierta el marco y el actor, ofrece una alternativa plausible en objetivo/ODS y falla en distinguir con precisión el rol institucional y el subtipo del movimiento.

### 5.3.2. Documento 2 — Comparación detallada

**Texto fuente**

> La movilización del 8M en Montevideo mostró la gran capacidad de organización del movimiento feminista. Miles marcharon y muchas más aguardaron para avanzar cuando el recorrido quedó chico; tras tres horas apenas se recorrió una cuadra. No importó: nadie se movió, se formaron grupos de compañeras de distintas edades y experiencias, y fueron una en el canto, en la consigna y en el puño alzado.
>
> Encabezaron esta marcha histórica las mujeres que desde hace tres años sostienen las ollas populares. Fue un reconocimiento al esfuerzo y sacrificio de mujeres humildes y fuertes que, movidas por la solidaridad, cumplieron un rol social que debería haber asumido el Estado y que, ante su ausencia, sostuvieron a miles de uruguayas y uruguayos con un plato de comida y más.
>
> Dos de las protagonistas de esta gesta conversaron con el medio y compartieron experiencias y aprendizajes, extendiendo un abrazo a todas las mujeres que, en todo el país, no dudaron en tender la mano y sostener a sus iguales.
>
> Gabriela Ríos es referente de las ollas en Bella Italia, un barrio golpeado por la pobreza. Sirve 555 porciones diarias de lunes a viernes. La red que encabeza cuenta con 11 ollas, cuando con insumos del MIDES llegaba a 29.
>
> Lourdes Gómez es referente del Barrio Municipal y su olla está en situación crítica: dejó de recibir insumos del MIDES y, por tomarse unos días para ir a Bella Unión, sus hijas no lograron servir la merienda; quedó suspendida de los insumos de la Intendencia y aguarda nueva visita para acceder nuevamente. Cientos de personas asisten tres días a la semana y los restantes sirven leche para niñas y niños.
>
> Las mujeres que llevan adelante una olla también son madres y responsables de tareas de cuidado. La hija pequeña de Gabriela casi se crió en la olla y aprendió sobre solidaridad. Lourdes relata que unió a su familia al trabajo: su esposo y sus hijas ayudaban a cocinar y limpiar cuando estaban sin trabajo.
>
> Sobre la motivación, Gabriela cuenta que siempre tuvo presente la solidaridad; veía necesidades en su barrio pero no sabía qué hacer. Con muchos niños y padres hurgadores que quedaron sin poder salir durante la pandemia, decidió organizar una olla popular que al 26 de marzo cumplió tres años.
>
> Lourdes se quedó sin trabajo y un amigo comenzó a acercarle insumos reunidos con compañeros; le propusieron poner una olla. Tenía experiencia en cocinar para mucha gente: es oficial pintora y albañil, trabajadora de la construcción y vinculada al SUNCA,

donde integró equipos que cocinaban para más de mil personas. Sus hijas la alentaron y dio el paso.

Ninguna de las dos tenía experiencia previa en organización barrial o comisiones vecinales; eran más bien cerradas con sus vecinos. La experiencia solidaria les enseñó mucho y lo cuentan con orgullo. Gabriela dice que antes no trataba con la gente pese a trabajar con público; hoy siente que la solidaridad le quitó una parte de su vida pero le dio otra mejor: ahora puede hablar, decidir y plantarse.

Un caso emblemático de la olla de Lourdes fue el de una vecina que jamás hablaba con nadie: se quedó sin trabajo, pidió asistir a la olla, fue recibida, acompañada e integrada al barrio. Muchas personas se acercaron y se pusieron a pelar verduras o limpiar; se formó mucha unión.

En su experiencia, las mujeres sostienen las ollas porque no es viable pagar una niñera con los salarios; la edad juega en contra para conseguir empleo después de los 50 y a las jóvenes les piden referencias que no tienen. Las distancias también dificultan: entre traslados y jornadas, casi no queda tiempo para atender a hijos e hijas. La mayoría de las mujeres con niñez a cargo son jefas de hogar.

Lo que queda es "apechugar" y buscar el plato de comida como sea. Las ollas se transformaron en espacios de contención y colaboración. Muchas mujeres que comenzaron asistiendo a alimentarse asumieron tareas en la olla y el cuidado de niños se volvió colectivo. Para Gabriela, formar un colectivo en el barrio "hace un montón"; Lourdes plantea que deben generarse oportunidades laborales barriales para el desarrollo de las mujeres.

Estas son las mujeres que el ministro de Desarrollo Social, Martín Lema, acusó de usar recursos del Estado de forma irregular. Enviaron jefes de oficinas, cargos políticos y no técnicos a inspeccionar, y luego, según denuncian, se utilizó información distorsionada para acusarlas de delitos.

Son mujeres que se sintieron violentadas una vez más por el Estado y lloraron de rabia e impotencia al ver sus nombres en titulares con acusaciones falsas. Durante tres años han sostenido, cuidado y tejido comunidad en sus barrios, buscando salidas colectivas a una crisis de hambre y desempleo.

Encabezar la marcha fue, para Gabriela, un orgullo: con todo lo trabajado, sintió que por lo menos en parte se las valoró al estar al frente.

Entre los cursos brindados por la Intendencia a estas militantes sociales estuvo "Fortalecidas", un programa para fortalecer la participación y el empoderamiento de mujeres y disidencias a nivel personal y colectivo, favoreciendo capacidades para incidir en sus vidas y entorno, promoviendo autonomía, autoestima y proyección.

Gabriela cuenta que no creía en el empoderamiento, pero entendió que no se puede ser "trapo de piso" de otra persona, que las mujeres valen y deben ser valoradas. Dice haber crecido como mujer y persona en estos tres años.

En 2023, primer año pospandemia, pese a anuncios de mejoras económicas, el hambre no cesó. La Intersocial Feminista denunció que las mujeres han cubierto retrocesos a costa de más explotación y precarización; las mujeres de ollas y merenderos sostienen barrios enteros. Esta respuesta popular y feminista expone la ausencia del Estado, que ya retiró apoyos a las ollas precisamente en marzo.

Cuando la crisis campea, el pueblo responde y las mujeres, en los territorios, demuestran su capacidad organizativa y de transformación social.

*El Popular*[2]

[2] https://elpopular.uy/las-mujeres-de-las-ollas-al-frente/

**Geográfica**

| Subclave | Esperado | Extraído |
|---|---|---|
| Región | SUR | SUR |
| Departamento | MONTEVIDEO | MONTEVIDEO |
| Localidad | NO CORRESPONDE | NO ESPECIFICA |

**Acciones**

| Subclave | Esperado | Extraído |
|---|---|---|
| Objetivo | Emancipatorios (superadores) | Reivindicativa |
| Matriz | Manifestación/Protesta | Manifestación/Protesta |
| Derivadas | Movilización | Marcha |

**Movimientos**

| Subclave | Esperado | Extraído |
|---|---|---|
| Tipo | Movimiento Consumo Alimentario (MCA) | Movimientos Mujeres Rurales |
| Nombre | OTROS | OTROS; Ollas Populares |
| Otros movimientos | mujeres que desde hace tres años sostienen las Ollas Populares | Ollas Populares |
| Mov. relacionados | Ollas en Bella Italia; Barrio Municipal | – |

**Institucional**

| Subclave | Esperado | Extraído |
|---|---|---|
| Tipo | N/C | Movimientos sociales |
| Nombre | – | Intersocial feminista |

**ODS**

| Subclave | Esperado | Extraído |
|---|---|---|
| Código | ODS2 | ODS5 |
| Descripción | Poner fin al hambre | Lograr la igualdad entre los géneros y empoderar a todas las mujeres y las niñas |

**Conclusión**

El modelo acierta el encuadre central: una movilización del 8M encabezada por mujeres vinculadas a las ollas populares, con correcta anclaje geográfico (SUR/Montevideo) y sin necesidad de precisar localidad. En **acciones**, "Manifestación/Protesta" y la derivada "Marcha" están bien; la discrepancia en el objetivo ("Emancipatorios" vs "Reivindicativa") es más de enfoque que de contenido: el texto permite leer tanto organización comunitaria y fortalecimiento (emancipatorio) como demanda al Estado ante el retiro de apoyos (reivindicativa). El desliz principal está en la tipificación del movimiento: clasifica "Movimientos Mujeres Rurales" cuando el caso refiere a un frente urbano-feminista ligado a ollas populares; aun así,

reconoce parcialmente al extraer “Ollas Populares”, pero omite referencias barriales (“Bella Italia”, “Barrio Municipal”) como relacionados. En lo **institucional**, añadir “Intersocial feminista” suma contexto aunque tiende a mezclar actor social con categoría institucional del esquema. En **ODS**, optar por ODS5 (igualdad de género) es defendible por el marco del 8M, pero el hilo del hambre y la asistencia alimentaria justificaría ODS2; un etiquetado múltiple o una prioridad condicionada por el lead resolvería la ambigüedad. En suma, el sistema acierta marco, geografía y forma de acción, y necesita mejorar la subtipificación del movimiento y la priorización objetivo/ODS según el foco narrativo del texto.

### 5.3.3. Documento 3 — Comparación detallada

#### Texto fuente

> Dos de las principales gremiales del sector agropecuario, la Asociación Rural del Uruguay (ARU) y la Federación Rural (FR) —integrantes de Campo Unido— solicitaron al Poder Ejecutivo una reducción de la alícuota de la Contribución Inmobiliaria Rural para los propietarios de padrones rurales, considerando que persisten las dificultades generadas por la sequía.
>
> El pedido fue dirigido al presidente de la República, Luis Lacalle Pou; a la ministra de Economía y Finanzas (MEF), Azucena Arbeleche; y al ministro de Ganadería, Agricultura y Pesca (MGAP), Fernando Mattos, expresidente de la ARU.
>
> En una carta conjunta, las gremiales expusieron: “Debido a los efectos del déficit hídrico en todo el territorio nacional, solicitamos una reducción de la alícuota de la Contribución Inmobiliaria Rural a los propietarios de padrones rurales establecida por el artículo 652 de la Ley Nº 15.809, de 8 de abril de 1986”.
>
> Se indicó además: “El incremento 421/022 establece que el valor real de los inmuebles para el año 2022 se determinará aplicando un coeficiente de 1,0995 a los valores reales de 2021. Considerando la tendencia a la baja del tipo de cambio nominal, el pago de la Contribución Inmobiliaria Rural en 2023 acaba siendo superior al 23 % respecto del pago del año anterior por igual concepto”.
>
> ARU y FR señalaron que, si bien en años anteriores se dispusieron beneficios para determinados departamentos, la situación de sequía alcanza hoy a todo el país; por ello, solicitan que la medida se extienda a todo el territorio.
>
> La carta fue firmada por Gonzalo Valdés Requena, presidente de la ARU, y por Martín Uría Shaw, presidente de la FR. Según consultas realizadas por *El Observador*, ambos ministerios acusaron recibo del planteo; por el momento no hubo respuesta y las gremiales aguardan una definición a corto plazo.
>
> *El Observador*[3]

#### Geográfica

| Subclave | Esperado | Extraído |
|---|---|---|
| Región | NO CORRESPONDE | NACIONAL |
| Departamento | – | – |
| Localidad | – | – |

[3] https://www.elobservador.com.uy/nota/lo-que-dos-gremiales-del-agro-pidieron-por-carta-a-lacalle-pou-y-a-dos-ministros-202339131819

**Acciones**

| Subclave | Esperado | Extraído |
|---|---|---|
| Objetivo | Propositiva | Reivindicativa |
| Matriz | Manifestación/Protesta | Vínculo con el Estado |
| Derivadas | Pronunciamientos | – |

**Movimientos**

| Subclave | Esperado | Extraído |
|---|---|---|
| Tipo | Movimiento Ruralista-Agronegocio (MRA) | Movimientos Ruralista-Agronegocio |
| Nombre | Asociación Rural del Uruguay | OTROS; Asociación Rural del Uruguay |
| Relacionados | Federación Rural del Uruguay | OTROS; Federación Rural |

**Institucional**

| Subclave | Esperado | Extraído |
|---|---|---|
| Tipo | Ministerios | Presidencia de la República |
| Otro tipo | MEF, MGAP | – |
| Nombre | Presidencia | Presidencia de la República |
| Acción de Estado | N/C | No corresponde |

**ODS**

| Subclave | Esperado | Extraído |
|---|---|---|
| Código | ODS8 | ODS13 |
| Descripción | Promover el crecimiento económico inclusivo y sostenible, el empleo y el trabajo decente para todos | Adoptar medidas urgentes para combatir el cambio climático y sus efectos |

**Conclusión**

El modelo capta bien el marco: un pedido formal de ARU y FR al Poder Ejecutivo para reducir la Contribución Inmobiliaria Rural por efectos de la sequía. La etiqueta geográfica Nacional resulta razonable al privilegiar el alcance de la solicitud sobre una localización física puntual. En **acciones**, la matriz Vínculo con el Estado describe mejor el hecho que Manifestación/Protesta; la diferencia de objetivo (Propositiva vs Reivindicativa) es matizable: el texto combina reclamo y propuesta concreta. Falta, no obstante, la derivada Pronunciamientos (la carta), que hubiese completado el encuadre. En **movimientos**, reconoce correctamente el tipo ruralista-agronegocio y a los actores (ARU y, como relacionado, FR), aunque el rótulo OTROS introduce ruido innecesario. En **institucional**, identificar Presidencia es parcialmente correcto pero incompleto: debió incorporar también a MEF y MGAP, explícitamente mencionados como destinatarios. En **ODS**, optar por ODS13 es defendible por el disparador (sequía), pero la finalidad inmediata es de alivio fiscal/productivo, por lo que ODS8 parece prioritaria con ODS13 como apoyo. En suma, el sistema acierta el marco, la escala y el canal institucional, y necesita afinar la granularidad institucional y la selección de objetivo/derivadas y ODS según el foco principal del texto.

### 5.3.4. Documento 4 — Comparación detallada

#### Texto fuente

> Mariela Martínez tuvo protagonismo nacional durante la pasada edición de Expo Prado, en setiembre, cuando la Asociación Rural del Uruguay decidió homenajearla —a ella y a otras mujeres del campo— y presentar la labor de una mujer, empresaria agropecuaria y madre duraznense, al país y al mundo.
>
> Hoy Mariela atraviesa un proceso médico, batallando otras luchas además de la vida en el campo. Este 8 de marzo sostiene su propuesta de salir adelante del mismo modo que lo hizo desde el trabajo rural.
>
> "Antes las movilizaciones por el Día de la Mujer se reducían específicamente al 8 de marzo. Ahora ocupa todo el mes y me alegra que así sea porque nos enteramos de nuevos proyectos, retomamos otros, siempre con el afán de superarnos; pero no podemos perder de vista el proyecto más importante, que es ser feliz. Tengo 60 años y viví siempre en el campo, y es ahí donde me siento en paz, en mi zona. Las mujeres que aún quedan son mayores de 50 años. Me pregunto: ¿qué hicimos mal que nos faltan dos generaciones de mujeres? ¿Será que les dijimos 'vayan a estudiar' y no les dijimos 'cuando se reciban, vuelvan y desarrollen un proyecto, grande o pequeño, de lo que aprendieron'?"
>
> Desde su residencia en el interior de Durazno agrega: "Nadie está obligado a vivir en el campo, pero créanme que es un lugar donde se puede parar para pensar. En este mes de marzo no quiero olvidarme de las mujeres de tercera edad, a las cuales muchas veces otras mujeres —hijas, nietas— las ponen en un hogar porque no hay quien las atienda. Lo mismo las niñas, que por distintos motivos no pueden crecer junto a sus padres. He visto de cerca una familia amiga de INAU que vive en el campo y esos niños son felices. Con tantos planes sociales que hay ahora, ¿no sería bueno preguntarles a esas abuelas si quieren ir a vivir al campo? Tendríamos que intentar algo por las que hablan bajito".
>
> *Su historia de vida.* Nació en el campo, donde la cría de ganado se imponía para vivir en familia. Se casó y se mudó a otro campo, a 3 kilómetros de su familia. Con su esposo emprendió en el rubro ganadero, criando vacas y ovejas. Adoptaron a Bruno y a Valentina, quienes concurrieron a escuelas rurales y luego continuaron sus estudios en Montevideo. Al enviudar, se hizo cargo del emprendimiento familiar y lo sostuvo como tal. Vacunos Aberdeen Angus y ovinos Highlander le permitieron el sustento mensual. "La ganadería me gusta y eso me sienta bien", dijo en la Expo Prado cuando su historia fue noticia. La ruralidad es su norte: allí se siente cómoda y plena, y desde allí ha sabido crear y criar una familia.
>
> *El Acontecer*[4]

#### Geográfica

| Subclave | Esperado | Extraído |
|---|---|---|
| Región | SUR | NORESTE |
| Departamento | MONTEVIDEO | DURAZNO |
| Localidad | NO CORRESPONDE | NO ESPECIFICA |

[4] https://elacontecer.com.uy/2023/03/08/mariela-martinez-mujer-rural-de-durazno-y-preocupada-por-la-tercera-edad-tendriamos-que-intentar-algo-por-las-que-hablan-bajito/

**Acciones**

| Subclave | Esperado | Extraído |
|---|---|---|
| Objetivo | Emancipatorios (superadores) | Propositiva |
| Matriz | Manifestación/Protesta | Entramados Socio-Culturales |
| Derivadas | Movilización | Eventos |

**Movimientos**

| Subclave | Esperado | Extraído |
|---|---|---|
| Tipo | Movimiento Mujeres Rurales (MMR) | Movimientos Mujeres Rurales |
| Nombre | OTROS; Mujeres rurales (SUTTA y UNATRA) | OTROS; Asociación Rural del Uruguay |

**Institucional**

| Subclave | Esperado | Extraído |
|---|---|---|
| Tipo | N/C | Otros |
| Otro tipo | – | Asociación Rural del Uruguay |
| Nombre | – | Asociación Rural del Uruguay |
| Acción de Estado | N/C | No corresponde |

**ODS**

| Subclave | Esperado | Extraído |
|---|---|---|
| Código | ODS5 | ODS5 |
| Descripción | Lograr la igualdad entre los géneros y empoderar a todas las mujeres y las niñas | Lograr la igualdad entre los géneros y empoderar a todas las mujeres y las niñas |

**Conclusión**

El modelo leyó el texto como perfil/relato de una mujer rural y no como crónica de protesta, y eso se refleja en varias decisiones. En **geografía**, aunque se aparta del "esperado" (SUR/Montevideo), la elección de Durazno es coherente con el foco biográfico del artículo. En **acciones**, el encuadre Entramados socio-culturales/Eventos y el objetivo Propositiva resultan plausibles para un homenaje/8M sin protesta explícita; "Manifestación/Protesta" y "Movilización" del esperado lucen menos ajustados al texto. En **movimientos**, acierta el tipo (Mujeres Rurales) pero desprioriza al actor principal al traer a ARU como nombre, cuando ARU aparece como entidad homenajeante y no como movimiento; esto también contamina la dimensión **institucional**, donde se mezcla un actor social con la categoría institucional del esquema. En **ODS**, ODS5 es consistente con el eje del texto (mujeres, 8M). En suma, el sistema acierta el tono (perfil propositivo y género) y el ODS, pero debe afinar la regionalización y, sobre todo, la selección del actor principal frente a entidades mencionadas como contexto.

# Capítulo 6

# Costos, métricas y viabilidad Operativa

En este capítulo se presenta un análisis de los costos y métricas operativas asociados al sistema de extracción y clasificación de noticias. Para ello, se desarrolló una suite de *scripts* que permiten recolectar datos históricos de ejecución, calcular promedios y proyectar el costo operativo de la infraestructura basada en modelos de lenguaje.

El procedimiento seguido consta de tres etapas principales. En primer lugar, se evaluó el flujo de entrada de información a partir de Google Alerts, considerando tanto el número de alertas recibidas como la cantidad de enlaces incluidos en cada una. En segundo lugar, se analizó la precisión del pipeline de clasificación, tomando como referencia una corrida experimental con integración a Pinecone y un esquema de RAG. Finalmente, a partir de las métricas de entrada y salida, se modelaron escenarios de costo operativo que permiten extrapolar el gasto mensual y anual, así como estimar el costo por caso relevante identificado.

El análisis que se presenta a continuación integra la perspectiva cuantitativa (volumen de alertas y enlaces) y la perspectiva cualitativa (tasa de relevancia y efectividad del *pipeline*). De este modo, se busca dimensionar el costo operativo del sistema y discutir su viabilidad y escalabilidad.

## 6.1. Métricas de entrada de alertas

Durante el período analizado, que abarca doce meses consecutivos, se recibieron en total **2.863 alertas**, que dieron lugar a **8.879 enlaces** (noticias). Estos valores permiten calcular los siguientes indicadores:

- **Promedio mensual**: aproximadamente **239 alertas por mes** y **740 enlaces por mes**.
- **Promedio diario**: considerando todos los días del calendario, se registraron en promedio **8,6 alertas diarias** y **27 enlaces diarios**.
- **Relación links/alerta**: cada alerta incluyó en promedio **3,1 enlaces**.

Estas métricas reflejan que el sistema opera de manera continua, con actividad prácticamente diaria. La densidad de enlaces por alerta es consistente a lo largo del año, oscilando

en torno a tres links por mensaje recibido. Este volumen de entrada constituye la base sobre la cual se calcula el costo de procesamiento, ya que cada enlace requiere un paso de *scraping* y posterior clasificación mediante el modelo de lenguaje. En términos operativos, un mes típico implica el procesamiento de cerca de **740 enlaces**, mientras que en un día cualquiera deben evaluarse alrededor de **27 enlaces**.

## 6.2. Rendimiento de clasificación

Para evaluar la efectividad del *pipeline* más allá del volumen de alertas y enlaces, la Tabla 6.1 sintetiza conteos y tasas clave (312 alertas; 966 enlaces), mientras que la Figura 6.1 visualiza la reducción progresiva. En términos de evaluación, el sistema alcanzó una cobertura (***recall***) del **91,2 %**, al recuperar 881 de los 966 enlaces disponibles. Todos los enlaces recuperados fueron procesados por el clasificador, y entre ellos, 547 fueron identificados como relevantes, lo que implica una precisión (***precision***) del **62,1 %**. Si se considera el total de enlaces desde el inicio, la proporción de relevantes representa un **56,6 %**. A nivel de alerta, el **69,6 %** (217/312) incluyó al menos un enlace clasificado como relevante.

Tabla 6.1: Resumen de métricas de procesamiento (denominador: links totales = 966).

| **Etapa** | **Conteo** | **% sobre total** | **Comentarios** |
|---|---|---|---|
| Alertas procesadas | **312** | — | Unidad de análisis alternativa |
| Links totales | **966** | **100,0 %** | Denominador común |
| Scraping exitoso | **881** | **91,2 %** | 85 links fallidos |
| Clasificados | **881** | **91,2 %** | Coincide con *Scraping exitoso* en esta corrida |
| Relevantes | **547** | **56,6 %** | Equivale a 62,1 % sobre los 881 con scraping exitoso |
| Alertas con $\geq 1$ relevante | **217/312** | **69,6 % (alertas)** | Métrica a nivel alerta |

Las tasas se calculan sobre 966 enlaces totales salvo que se indique lo contrario.

La Figura 6.1 resume gráficamente el embudo con denominador común (*links totales* = 966).
*Nota.* En esta corrida, **Clasificados** coincide con **Scraped OK**; en otras ejecuciones pueden diferir si hubiera fallas de inferencia o descartes por formato.

## 6.3. Comparativa de modelos (precios por tokens)

En este trabajo se emplearon dos modelos provistos por **OpenAI**: *GPT-4o* y *GPT-4o-mini*. El primero se utiliza como modelo de referencia por su mayor capacidad de razonamiento y comprensión contextual, mientras que la versión *mini* ofrece una alternativa más económica para tareas de extracción a costo de una menor capacidad de razonamiento.

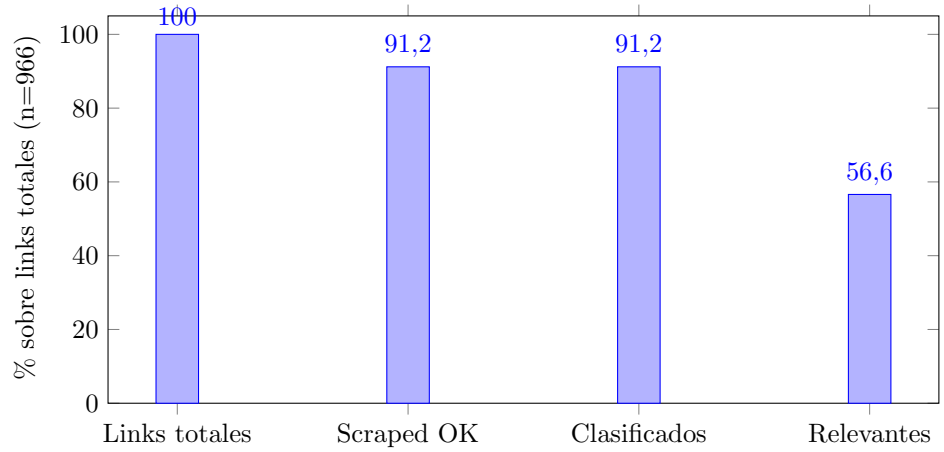



Figura 6.1: Embudo de efectividad del pipeline con denominador común (links totales = 966). Relevantes: 547/966 = 56,6 % (equivale a 547/881 = 62,1 % sobre *scraping* exitoso).

La arquitectura implementada permite alternar dinámicamente entre ambos modelos a través de una variable de entorno, lo que provee un mecanismo para elegir el modelo apropiado en base a las necesidades del OCAU.

La Tabla 6.2 presenta los precios por millón de tokens de entrada y salida para los modelos GPT-4o y GPT-4o mini reportados por OpenAI en 2025.

| Modelo | Input (USD / $10^6$ tokens) | Output (USD / $10^6$ tokens) |
|---|---|---|
| GPT-4o | 2,50 | 10,00 |
| GPT-4o mini | 0,15 | 0,60 |

Tabla 6.2: Precios oficiales por tokens (OpenAI, 2025).

## 6.4. Modelo de costos y escenarios

Se modela el costo operativo suponiendo un consumo promedio de **13.000 tokens por enlace**, con una proporción de **80 %** tokens de entrada y **20 %** de salida. Sea $t_{\text{in}} = 10{,}400$ y $t_{\text{out}} = 2{,}600$. El costo por enlace (USD) para un modelo con precios $(p_{\text{in}}, p_{\text{out}})$ por millar de tokens se define como:

$$\text{costo\_link} \;=\; \frac{t_{\text{in}}}{1000} \cdot p_{\text{in}} \;+\; \frac{t_{\text{out}}}{1000} \cdot p_{\text{out}}.$$

### Costos por modelo (80/20, 13k)

Para **GPT-4o**: $p_{\text{in}} = 0{,}0025$, $p_{\text{out}} = 0{,}01$.

$$\text{costo\_link}^{(4o)} = 10{,}4 \cdot 0{,}0025 + 2{,}6 \cdot 0{,}01 = \mathbf{0{,}052}.$$

Para **GPT-4o mini**: $p_{\text{in}} = 0{,}00015$, $p_{\text{out}} = 0{,}0006$.

$$\text{costo\_link}^{(4o\_mini)} = 10{,}4 \cdot 0{,}00015 + 2{,}6 \cdot 0{,}0006 = \mathbf{0{,}00312}.$$

## Escenarios operativos

Se reportan dos escenarios:

- **Escenario A:** procesamiento integral del 100 % de los enlaces.
- **Escenario B:** cálculo acotado al **56 %** de válidas (valor conservador observado sobre el total).

Se incluyen además referencias de volumen de enlaces:

- Promedio **mensual**: $L_m \approx 740$ enlaces.
- Promedio **anual**: 8,879 enlaces.
- Promedio **diario**: $\bar{L}_d \approx 27$ enlaces.

**GPT-4o.**

$$\text{costo\_mes}^{(A)} = 740 \times 0{,}052 = \mathbf{38{,}48}; \quad \text{costo\_año}^{(A)} = 8{,}879 \times 0{,}052 = \mathbf{461{,}71};$$

$$\text{costo\_día}^{(A)} = 27 \times 0{,}052 = \mathbf{1{,}40}.$$

Con 56 %:

$\text{costo\_mes}^{(B)} = 0{,}56{\times}38{,}48 = \mathbf{21{,}55}$; $\text{costo\_año}^{(B)} = 0{,}56{\times}461{,}71 = \mathbf{258{,}56}$; $\text{costo\_día}^{(B)} \approx \mathbf{0{,}79}$.

**GPT-4o mini.**

$$\text{costo\_mes}^{(A)} = 740 \times 0{,}00312 = \mathbf{2{,}31}; \quad \text{costo\_año}^{(A)} = 8{,}879 \times 0{,}00312 = \mathbf{27{,}70};$$

$$\text{costo\_día}^{(A)} = 27 \times 0{,}00312 \approx \mathbf{0{,}084}.$$

Con 56 %:

$\text{costo\_mes}^{(B)} = 0{,}56{\times}2{,}31 \approx \mathbf{1{,}29}$; $\text{costo\_año}^{(B)} = 0{,}56{\times}27{,}70 \approx \mathbf{15{,}51}$; $\text{costo\_día}^{(B)} \approx \mathbf{0{,}047}$.

## Costo por caso relevante

Considerando relevancia $R$ sobre el total de procesados, el costo esperado por relevante bajo el escenario integral es costo_link/$R$.

- **GPT-4o**: con $R = 0{,}56$, $0{,}052/0{,}56 \approx \mathbf{0{,}093}$ USD/relevante.
- **GPT-4o mini**: con $R = 0{,}56$, $0{,}00312/0{,}56 \approx \mathbf{0{,}0056}$ USD/relevante.

**Resumen comparativo de costos**

| Modelo | Costo/enlace | Mes (100 %) | Año (100 %) | Mes (56 %) | Año (56 %) |
|---|---|---|---|---|---|
| GPT-4o | 0,052 | 38,48 | 461,71 | 21,55 | 258,56 |
| GPT-4o mini | 0,00312 | 2,31 | 27,70 | 1,29 | 15,51 |

Tabla 6.3: Costos operativos (USD) con 13.000 tokens/enlace (80/20).

### 6.4.1. Costos durante la fase de investigación y desarrollo

Durante la fase de investigación y exploración de alternativas tecnológicas descrita en el Capítulo 3, se incurrió en costos asociados a la evaluación de diferentes proveedores de modelos de lenguaje.

En la sección 3.1, se emplearon variantes de Mistral-7B y Llama mediante Hugging Face Inference Providers[1], un servicio de inferencia administrado que, aunque utiliza modelos de código abierto como base, requiere pago por uso. Esta exploración inicial tuvo un costo aproximado de USD 10, período durante el cual se evaluó la viabilidad de estos modelos para las tareas de extracción estructurada del dominio del OCAU.

Como se documentó en la sección 3.1.6, los resultados obtenidos con estos modelos presentaron limitaciones significativas: menor adherencia al formato estructurado, dificultad para mapear enumeraciones rígidas de Pydantic, y mayor variabilidad en casos ambiguos. Estos hallazgos motivaron la transición hacia modelos de OpenAI.

La fase de investigación y desarrollo con OpenAI API (GPT-4o y GPT-4o-mini) representó un costo aproximado de USD 40. Este período incluyó:

- Iteración sobre arquitecturas de prompts y esquemas Pydantic
- Integración y prueba de herramientas MCP para validación de dominio
- Evaluación comparativa de GPT-4o vs GPT-4o-mini en diferentes extractores
- Ajuste de parámetros de clasificación y relevancia

**Costo total: USD 50** (10 + 40), inversión que permitió validar la arquitectura final y confirmar la viabilidad técnica y económica del sistema antes de su operación continua.

Esta inversión inicial fue fundamental para tomar decisiones arquitectónicas informadas que se materializan en los costos operativos proyectados presentados en la sección 6.4.

### 6.4.2. Reflexión Final: Justificación de Costos Operativos

La decisión de utilizar servicios externos de modelos de lenguaje (OpenAI API) se fundamenta en un análisis de costo-beneficio que evaluó distintas alternativas exploradas durante el desarrollo del proyecto.

Como se documentó en el Capítulo 3, los modelos de código abierto (Mistral-7B, Llama) mostraron limitaciones significativas para la extracción estructurada.

[1] Documentación oficial: https://huggingface.co/docs/inference-providers/en/index

Alternativamente, se evaluó utilizar los *clusters* de la Facultad para hospedar modelos. Sin embargo, el análisis costo-tiempo-beneficio reveló implicancias operativas considerables: trabajo de configuración y *fine-tuning*, gestión de infraestructura, logística de acceso para el cliente final (OCAU), y mantenimiento. Estos costos indirectos representaban una inversión considerablemente mayor que el costo monetario del uso de OpenAI presentado en este capítulo.

Frente a estas alternativas, los servicios de OpenAI presentaron ventajas decisivas:

- **Calidad del resultado**: Adherencia consistente al formato estructurado, reducción drástica de errores y resultados satisfactorios.
- **Costos justificadamente asumibles**: Dado el tipo de proyecto y la escala de información manejada, la relación costo-beneficio que estas herramientas presentan hace que su adopción sea plenamente justificable.
- **Simplicidad operativa**: Elimina la complejidad de gestionar infraestructura de modelos.

**Conclusión:**

La evaluación reveló que el uso de servicios externos de OpenAI se justifica plenamente al ponderar:

1. La calidad insuficiente de modelos libres para este dominio específico.
2. El costo-oportunidad del trabajo adicional requerido para infraestructura propia.
3. La relación favorable entre costo monetario y beneficio operativo dado el volumen de información procesada.

Esta decisión arquitectónica prioriza la efectividad, mantenibilidad y viabilidad operativa del sistema, garantizando resultados confiables.

# Capítulo 7

# Conclusiones y Trabajo Futuro

El presente trabajo logró desarrollar e implementar un sistema automatizado de extracción de información a partir de noticias sobre movimientos socioterritoriales rurales en Uruguay. El sistema, construido mediante una arquitectura modular que integra técnicas de procesamiento de lenguaje natural, recuperación de información y LLMs, demostró ser capaz de procesar de forma continua el flujo de alertas proveniente de Google Alerts y extraer información estructurada según las cinco dimensiones definidas por el OCAU: geografía, acciones, movimientos, instituciones y ODS.

## 7.1. Logros principales

El sistema implementado alcanzó los siguientes resultados concretos:

- **Automatización efectiva del flujo de trabajo**: *Pipeline* completamente automatizado desde la ingesta de alertas hasta la generación de datos estructurados, sin intervención manual en la clasificación ni en la extracción inicial. Procesa ∼740 enlaces mensuales con una tasa de identificación de contenido relevante del **56,6 %**.
- **Integración de validaciones y recuperación especializada**: La incorporación de MCP para verificaciones geográficas, semánticas y de consistencia, junto con la búsqueda híbrida en Pinecone, incrementó significativamente la precisión en el reconocimiento de entidades uruguayas y en la clasificación de relevancia mediante RAG.
- **Viabilidad económica demostrada**: Costos anuales de **USD 15,51** con GPT-4o mini (o **USD 258,56** con GPT-4o), con una relación costo–beneficio ampliamente superior al procesamiento manual equivalente.

### 7.1.1. Evaluación crítica y limitaciones

A pesar de los logros alcanzados, el trabajo enfrentó varias limitaciones que merecen análisis:

- **Ambigüedad inherente del dominio**: Las categorías del OCAU presentan fronteras difusas que generan discrepancias incluso entre anotadores humanos. Conceptos

como “resistencia” vs “reivindicativa” o “marcha” vs “manifestación” admiten interpretaciones múltiples válidas dentro del contexto provisto por el OCAU, lo que limita la posibilidad de alcanzar concordancia perfecta.

- **Complejidad en la extracción institucional**: El modelo presenta dificultades para distinguir entre actores sociales y entidades estatales cuando ambos aparecen en el mismo texto, tendiendo a confundir el rol de cada actor en la noticia.

## 7.2. Trabajo Futuro

Los resultados obtenidos abren múltiples líneas de desarrollo que pueden mejorar el sistema y expandir sus capacidades.

### 7.2.1. Mejoras generales

A continuación se presentan una serie de mejoras generales que vale la pena considerar a futuro:

- **Extensibilidad del modelo de dominio**
  Habilitar la incorporación ágil de nuevas entidades (p. ej., instituciones, movimientos, tipos de acciones), sin cambios disruptivos en la arquitectura ni en los extractores. Esto permitiría al OCAU adaptar el sistema a medida que evoluciona su marco conceptual de análisis.

- **Criterios de extracción y relevancia configurables**
  Permitir ajustar y versionar las reglas de extracción (normalización, validaciones MCP) y los umbrales de relevancia (búsqueda híbrida, filtros, *scoring*) mediante configuración declarativa, sin necesidad de modificar código.

- **Portabilidad a otros dominios**
  Adaptar el sistema para su uso en otros contextos de monitoreo social, manteniendo la estructura general del proceso (captura de noticias, clasificación, extracción de información estructurada) pero ajustando las categorías y criterios de análisis según las necesidades específicas.

- **Refinamiento de prompts con casos de borde**: Incorporar ejemplos específicos de casos ambiguos identificados durante la evaluación para mejorar la capacidad del modelo de manejar situaciones fronterizas entre categorías.

- **Dashboard interactivo y herramientas de gestión**
  Evolucionar el *dashboard* actual hacia una plataforma que combine visualización analítica con capacidades operativas. Este *dashboard* avanzado debería incluir:

  - **Edición y retroalimentación**: Interfaz para revisar y corregir las extracciones automáticas, generando un corpus de correcciones que permita el aprendizaje continuo del sistema y la identificación de patrones de error recurrentes.
  - **Gestión dinámica de filtros**: Herramientas para agregar, modificar o desactivar *keywords* utilizadas en la etapa de clasificación, permitiendo ajustar la sensibilidad del sistema sin intervención técnica.

### 7.2.2. Integración de interacción humano–computadora

Una línea complementaria de trabajo futuro apunta a incorporar interacción colaborativa entre humanos y el sistema, de modo que los analistas puedan participar activamente en el flujo de procesamiento en lugar de depender exclusivamente de decisiones automáticas. Esta expansión transformaría el pipeline en un proceso asistido, donde el modelo se ocupa de las tareas más tediosas y el usuario conserva la supervisión y decisión final.

Entre las funcionalidades propuestas se encuentran:

- **Inserción manual de noticias:** posibilidad de agregar una noticia específica mediante su URL para procesarla dentro del flujo general, incluso si no fue capturada automáticamente.
- **Eliminación de resultados inadecuados:** capacidad de borrar o marcar como erróneas las salidas del pipeline que no reflejen correctamente el contenido de la noticia.
- **Selección entre sugerencias múltiples:** mostrar varias categorías o interpretaciones sugeridas por el modelo para cada dimensión estructurada, permitiendo que el usuario elija la más adecuada según su criterio experto.
- **Supervisión asistida:** rediseñar el flujo para que el sistema presente la información procesada y las alternativas posibles, reduciendo la carga cognitiva del analista y fomentando un entorno de colaboración humano–máquina.

Estas funcionalidades fortalecerían la capacidad del sistema para combinar la eficiencia de la automatización con el criterio contextual humano, mejorando tanto la precisión de los resultados como la confianza en el proceso de análisis.

## 7.3. Reflexión Final

El desarrollo de este proyecto permitió consolidar y aplicar los conocimientos adquiridos durante toda la carrera, desde conceptos fundamentales de ingeniería de software hasta tecnologías avanzadas de procesamiento de lenguaje natural. El proceso de construcción del sistema, con sus múltiples iteraciones y desafíos técnicos, constituyó una experiencia formativa integral que trasciende lo puramente académico.

Trabajar con el OCAU brindó la oportunidad de contribuir a una iniciativa de relevancia social. Esperamos que esta herramienta resulte de utilidad y sea empleada de manera satisfactoria en sus labores de monitoreo y análisis. La posibilidad de que el sistema desarrollado pueda aportar valor al trabajo del observatorio representa una motivación adicional que enriqueció todo el proceso de desarrollo.

# Referencias


Barbosa, R. (2025, May). Building ai-powered document generation with box mcp and pydantic ai. *Box Developer Blog (Medium)*. Descargado de https://medium.com/box-developer-blog/building-ai-powered-document-generation-with-box-mcp-and-pydantic-ai-48775b18ae32

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., ... Liang, P. (2022). *On the opportunities and risks of foundation models.* Descargado de https://arxiv.org/abs/2108.07258

Box, Inc. (2025). *Box ai developer zone.* https://developer.box.com/ai-dev-zone/.

Chinchor, N., y Robinson, P. (1998). Appendix E: MUC-7 named entity task definition (version 3.5). En *Seventh message understanding conference (MUC-7): Proceedings of a conference held in fairfax, virginia, April 29 - may 1, 1998.* Descargado de https://aclanthology.org/M98-1028/

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., ... Stoyanov, V. (2020). *Unsupervised cross-lingual representation learning at scale.* Descargado de https://arxiv.org/abs/1911.02116

Cowie, J., y Lehnert, W. (1996). Information extraction. *Communications of the ACM*, *39*(1), 80–91. doi: 10.1145/234173.234209

Crawl4AI Project. (2025). *Llm strategies - crawl4ai documentation.* https://docs.crawl4ai.com/extraction/llm-strategies/.

Devlin, J., Chang, M.-W., Lee, K., y Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. En *Proceedings of naacl-hlt* (pp. 4171–4186).

Frederking, R., Hovy, E., y Wilks, Y. (1991). *Cross-lingual information extraction and automated text summarization* (Inf. Téc.). Carnegie Mellon University. Descargado de http://www.cs.cmu.edu/~ref/mlim/chapter3.html

Grishman, R. (2003). Information extraction. En R. Dale, H. Moisl, y H. Somers (Eds.), *The handbook of natural language processing.* Marcel Dekker.

Grishman, R., y Sundheim, B. (1996). Message Understanding Conference- 6: A brief history. En *COLING 1996 volume 1: The 16th international conference on computational linguistics.* Descargado de https://aclanthology.org/C96-1079/

Gulab, R. (2025, June). Extracting structured data (like emails or job info) using crawl4ai, llms, and pydantic. *Medium*. Descargado de https://medium.com/%40rajratangulab.more/extracting-structured-data-like-emails-or-job-info-using-crawl4ai-llms-pydantic-390632f87e70

Johnson, J., Douze, M., y Jégou, H. (2017). *Billion-scale similarity search with gpus.* Descargado de https://arxiv.org/abs/1702.08734

Jurafsky, D., y Martin, J. H. (2025). *Speech and language processing* (3rd (online draft) ed.). Descargado de https://web.stanford.edu/~jurafsky/slp3/

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., y Iwasawa, Y. (2023). *Large language models are zero-shot reasoners.* Descargado de https://arxiv.org/abs/2205.11916

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... Kiela, D. (2021). *Retrieval-augmented generation for knowledge-intensive nlp tasks.* Descargado de https://arxiv.org/abs/2005.11401

Microsoft Learn. (2025a). *Azure openai in azure ai foundry models rest api (preview).* https://learn.microsoft.com/en-us/azure/ai-foundry/openai/reference-preview.

Microsoft Learn. (2025b). *Extract entities using azure openai structured outputs mode.* https://learn.microsoft.com/en-us/azure/developer/ai/how-to/extract-entities-using-structured-outputs.

Microsoft Learn. (2025c). *How to use structured outputs with azure openai.* https://learn.microsoft.com/en-us/azure/ai-foundry/openai/how-to/structured-outputs.

Novotny, A. (2025). *Create an agentic workflow using box, pydantic, and openai (with mcp).* YouTube. Descargado de https://www.youtube.com/watch?v=-290uDt3SWU

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., y Riedel, S. (2019). *Language models as knowledge bases?* Descargado de https://arxiv.org/abs/1909.01066

Ramshaw, L. A., y Marcus, M. P. (1999). Text chunking using transformation-based learning. En M. Stevenson y A. Way (Eds.), *Natural language processing using very large corpora* (pp. 157–176). Springer. doi: 10.1007/978-94-017-2390-9_10

Rogers, A., Kovaleva, O., y Rumshisky, A. (2020). *A primer in bertology: What we know about how bert works.* Descargado de https://arxiv.org/abs/2002.12327

Salinas, A., y Morstatter, F. (2024). The butterfly effect of altering prompts: How small changes and jailbreaks affect large language model performance. En *Findings of the association for computational linguistics: Acl 2024* (pp. 4561–4577). Association for Computational Linguistics. Descargado de https://aclanthology.org/2024.findings-acl.275

Sarawagi, S. (2008). Information extraction. *Foundations and Trends in Databases*, *1*(3), 261–377. doi: 10.1561/1900000003

Sclar, M., Choi, Y., Tsvetkov, Y., y Suhr, A. (2024). *Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting.* Descargado de https://arxiv.org/abs/2310.11324

Tjong Kim Sang, E. F., y Buchholz, S. (2000). Introduction to the conll-2000 shared task: Chunking. En *Proceedings of conll-2000 and lll-2000* (pp. 127–132).

unclecode. (2025). *Crawl4ai: Open-source ai-ready web crawler.* https://github.com/unclecode/crawl4ai.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., ... Le, Q. V. (2022a). *Finetuned language models are zero-shot learners.* Descargado de https://arxiv.org/abs/2109.01652

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., ... Le, Q. V. (2022b). *Finetuned language models are zero-shot learners.* Descargado de https://arxiv.org/abs/2109.01652

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., ... Raffel, C. (2021). *mt5: A massively multilingual pre-trained text-to-text transformer.* Descargado de https://arxiv.org/abs/2010.11934

Zanini, A. (2025, Aug). Pydantic ai with bright data's web mcp for agents with data access. *Bright Data Blog*. Descargado de https://brightdata.com/blog/ai/pydantic-ai-with-web-mcp

Zhang, H., Liu, J., Zhu, Z., Zeng, S., Sheng, M., Yang, T., ... Wang, Y. (2024). *Efficient and effective retrieval of dense-sparse hybrid vectors using graph-based approximate nearest neighbor search.* Descargado de https://arxiv.org/abs/2410.20381

# Glosario

**Agente** Componente que actúa de forma autónoma u orquestada para resolver tareas; puede invocar herramientas, razonar con contexto y coordinar pasos de un *pipeline*.

**Backend** Parte del sistema que gestiona la lógica de negocio, procesamiento de datos y comunicación con bases de datos. Proporciona servicios que consume el *frontend* a través de APIs.

**Backoff** Estrategia de espera progresiva entre reintentos de operaciones fallidas para evitar sobrecarga de servicios externos.

**Backtracking** Técnica algorítmica que revierte decisiones previas para encontrar soluciones alternativas; utilizada para verificar casos previamente clasificados comparando la salida del sistema con el esquema objetivo.

**Bases de datos vectoriales** Sistemas diseñados para almacenar y consultar representaciones vectoriales (*embeddings*) con métricas de similitud; útiles para recuperación semántica.

**bcrypt** Algoritmo de hashing criptográfico diseñado específicamente para el almacenamiento seguro de contraseñas, que incorpora un factor de costo configurable para proteger contra ataques de fuerza bruta.

**Broker** Intermediario de mensajes que gestiona el intercambio de mensajes entre diferentes partes del sistema; en este proyecto se utiliza Redis como *broker* para Celery.

**Bundler** Herramienta que empaqueta y optimiza archivos de código fuente (JavaScript, CSS, imágenes) en uno o más archivos optimizados para producción. Ejemplos: Vite, Webpack.

**Caché** Almacenamiento temporal de datos frecuentemente accedidos para reducir tiempo de carga y llamadas al servidor. Mejora el rendimiento manteniendo copias locales de información.

**Celery** Cola de tareas para Python que permite ejecución asíncrona, programación periódica (con Celery Beat) y reintentos con políticas configurables.

**Chain-of-Thought (CoT)** Técnica de *prompting* que induce al modelo a razonar paso a paso antes de entregar una respuesta final, mejorando el rendimiento en tareas que requieren lógica o interpretación contextual.

**Checkpointing** Proceso de guardar el estado del sistema en puntos específicos para permitir recuperación o reanudación de procesos interrumpidos.

**Clasificación** Tarea de asignar etiquetas a documentos o fragmentos según criterios definidos (p. ej., relevancia temática o pertinencia).

**Clustering** Agrupamiento. Técnica de análisis de datos que agrupa elementos similares sin supervisión previa.

**Componente** Pieza reutilizable y autocontenida de interfaz de usuario que encapsula su propia lógica, presentación y comportamiento. En React, pueden ser funcionales o de clase.

**Dashboard** Interfaz visual que presenta información clave y métricas del sistema de forma consolidada, permitiendo monitoreo y toma de decisiones rápidas.

**Dense vectors** Vectores densos. Representaciones vectoriales de alta dimensionalidad con valores distribuidos, generados por modelos como BERT o GPT para capturar significado semántico.

**Embedding** Representación numérica densa de un texto que preserva similitud semántica; se usa para búsqueda y *clustering*.

**Endpoint** Punto de acceso específico de una API que responde a peticiones HTTP en una URL determinada, representando un recurso o acción del sistema.

**FastAPI** Framework web para construir APIs en Python con tipado estático, validación automática y soporte asíncrono.

**FastMCP** Implementación/librería para exponer herramientas y contexto vía Model Context Protocol (MCP) de forma ligera.

**Feature Flag** Técnica de desarrollo que permite habilitar o deshabilitar funcionalidades del sistema de forma dinámica sin necesidad de redesplegar la aplicación, facilitando el despliegue continuo y las pruebas A/B.

**Framework** Marco de trabajo. Estructura conceptual y tecnológica de soporte para el desarrollo de software que proporciona funcionalidades base reutilizables.

**Frontend** Parte visible de una aplicación web con la que los usuarios interactúan directamente. Se encarga de la presentación de datos y la interfaz de usuario.

**Function calling** Llamada de funciones. Mecanismo que permite a los modelos de lenguaje invocar funciones externas para ampliar sus capacidades más allá del texto.

**Grafo** Estructura compuesta por nodos y aristas; se usa para modelar flujos (*pipelines*) o dependencias entre componentes.

**Hashing** Proceso criptográfico unidireccional que transforma datos (como contraseñas) en una cadena de longitud fija, imposible de revertir al valor original.

**Herramienta (tool)** Función o servicio externo que el sistema puede invocar (p. ej., buscador, validador, base de datos) para ampliar capacidades del LLM.

**Hook** En React, funciones especiales que permiten usar estado y otras características de React en componentes funcionales. Ejemplos incluyen `useState`, `useEffect`, y *custom hooks* creados por el desarrollador.

**Idempotencia** Propiedad que garantiza que realizar la misma operación múltiples veces produce el mismo resultado, fundamental para sistemas de procesamiento automatizado.

**LangChain** Framework para construir aplicaciones con LLMs; provee cadenas, herramientas, *Runnables* y utilidades de orquestación.

**LangGraph** Extensión orientada a grafos para modelar agentes y flujos con estado y bucles de control en aplicaciones LLM.

**Model Context Protocol (MCP)** Protocolo que estandariza cómo un agente/LLM descubre e invoca herramientas externas y accede a contexto de manera segura.

**MongoDB** Base de datos NoSQL orientada a documentos que almacena datos en formato JSON/BSON, ofreciendo flexibilidad de esquema y alta escalabilidad.

**Mutation** En el contexto de React Query, operación que modifica datos en el servidor (crear, actualizar, eliminar). Se diferencia de una *query* que solo lee datos.

**Nodo** Unidad elemental de un grafo o *pipeline*; encapsula una operación (p. ej., un extractor o un clasificador).

**Normalización** Proceso de estandarizar datos (nombres, fechas, ubicaciones) para reducir ambigüedad y mejorar la consistencia.

**One-shot learning** Aprendizaje con un ejemplo. Variante de *few-shot* con un único ejemplo que ilustra el comportamiento esperado.

**Paginación** Técnica que divide grandes conjuntos de resultados en páginas más pequeñas y manejables, mejorando el rendimiento y la experiencia del usuario.

**Parsing** Análisis sintáctico. Proceso de analizar una cadena de símbolos conforme a las reglas de una gramática para extraer estructura.

**Pinecone** Servicio de base de datos vectorial administrada, usado para indexar y consultar *embeddings* a gran escala.

**Pipeline** Flujo encadenado de pasos (extracción, validación, almacenamiento, etc.) que transforma entradas en salidas útiles.

**Precisión** Métrica de evaluación que indica la proporción de elementos recuperados que son verdaderamente relevantes. Se calcula como el cociente entre los verdaderos positivos (VP) y la suma de verdaderos positivos y falsos positivos (FP):

$$\text{Precisión} = \frac{VP}{VP + FP}$$

En este contexto, mide la calidad de los enlaces clasificados como relevantes por el sistema.

**Procesamiento de Lenguaje Natural (PLN)** Área de la informática y la lingüística computacional que diseña métodos para que las computadoras analicen, comprendan y generen lenguaje humano. Abarca etapas como preprocesamiento y representación (tokens, subpalabras, *embeddings*) y tareas como clasificación de textos, etiquetado de secuencias, extracción de información, respuesta a preguntas y generación controlada.

**Prompt** Consigna o instrucción que guía la salida de un modelo de lenguaje; puede incluir ejemplos, formato deseado y restricciones.

**Prompting** Técnica de guiar modelos de lenguaje mediante instrucciones específicas, ejemplos o formatos deseados para obtener salidas estructuradas.

**Pydantic** Librería de validación y modelado de datos en Python basada en anotaciones de tipo; genera y aplica esquemas en tiempo de ejecución.

**Query** En el contexto de React Query, operación de lectura que obtiene datos del servidor. Se diferencia de una *mutation* que modifica datos.

**Rate limit** Límite de velocidad. Restricción sobre la frecuencia de solicitudes permitidas a un servicio para prevenir sobrecarga.

**Recall** Métrica de evaluación que mide la capacidad del sistema para recuperar los elementos relevantes. Se define como el cociente entre los verdaderos positivos (VP) y la suma de verdaderos positivos y falsos negativos (FN):

$$\text{Recall} = \frac{VP}{VP + FN}$$

En este trabajo, representa el porcentaje de enlaces relevantes efectivamente recuperados por el *pipeline*.

**Recall@10** Métrica de evaluación en recuperación de información que mide la proporción de documentos relevantes presentes entre los 10 primeros resultados devueltos por el sistema..

**Reintentos** Mecanismo de repetición automática de operaciones fallidas con políticas configurables de espera y número máximo de intentos.

**Retrieval-Augmented Generation (RAG)** Patrón que combina recuperación de contexto (p. ej., búsqueda vectorial) con generación de texto para producir respuestas con apoyo en evidencia.

**Router** Componente que agrupa y organiza endpoints relacionados en una API, facilitando la modularización y el mantenimiento del código.

**Routing** Sistema que gestiona la navegación entre diferentes vistas o páginas de una aplicación web, mapeando URLs a componentes específicos.

**Schema** Esquema. Estructura que define la organización y validación de datos, especificando tipos, campos obligatorios y restricciones.

**Schema validation** Validación de esquemas. Verificación automática de que los datos cumplen con una estructura predefinida antes de su procesamiento.

**Scraping** Extracción automática de contenido desde páginas web; suele incluir limpieza de HTML, selección de elementos y normalización.

**Sparse vectors** Vectores dispersos. Representaciones vectoriales con mayoría de valores cero, útiles para coincidencias léxicas exactas de nombres propios y terminología técnica.

**Structured outputs** Salidas estructuradas. Resultados generados en formatos predefinidos y validables (como JSON) que facilitan el procesamiento automático posterior.

**Template** Plantilla. Modelo predefinido que se utiliza como base para generar contenido con estructura consistente.

**Timeout** Tiempo límite. Período máximo permitido para completar una operación antes de considerarla fallida.

**Token** Unidad mínima de procesamiento para el modelo (subpalabra, palabra o símbolo) empleada en entrenamiento e inferencia.

**Top-k** Los k mejores resultados. Técnica de recuperación que retorna los k elementos más relevantes según una métrica de similitud.

**Transformer** Familia de modelos que usa mecanismos de atención para modelar dependencias de largo alcance; base de la mayoría de LLMs modernos.

**Umbral** Valor de corte usado para decidir entre clases o filtrar resultados (p. ej., similitud mínima).

**Unicode** Estándar internacional de codificación de caracteres que permite representar texto de prácticamente todos los sistemas de escritura del mundo..

**Upsert** Operación que inserta o actualiza datos según existan o no previamente, garantizando idempotencia en el almacenamiento.

**Uvicorn** Servidor web ASGI ligero y de alto rendimiento para aplicaciones Python asíncronas, especialmente optimizado para frameworks como FastAPI.

**Ventana de contexto** Longitud máxima de tokens que el modelo considera simultáneamente como entrada.

**Workflow** Flujo de trabajo. Secuencia automatizada de tareas y procesos que coordina el avance de documentos a través del sistema.

**Zero-shot learning** Aprendizaje sin ejemplos. Capacidad de realizar tareas sin ejemplos específicos de entrenamiento, basándose únicamente en instrucciones textuales.

# Lista de Acrónimos

**API** Application Programming Interface.

**ASGI** Asynchronous Server Gateway Interface.

**CORS** Cross-Origin Resource Sharing.

**CRF** Conditional Random Fields.

**CRUD** Create, Read, Update, Delete.

**CSS** Cascading Style Sheets.

**HTML** HyperText Markup Language.

**HTTP** Hypertext Transfer Protocol.

**IE** Extracción de Información.

**JSON** JavaScript Object Notation.

**JWT** JSON Web Token.

**LLM** Modelo de Lenguaje de Gran Escala.

**MCP** Model Context Protocol.

**NLP** Natural Language Processing.

**OAuth2** Open Authorization 2.0.

**OCAU** Observatorio de la Cuestión Agraria del Uruguay.

**ODS** Objetivos de Desarrollo Sostenible.

**PLN** Procesamiento de Lenguaje Natural.

**RAG** Retrieval-Augmented Generation.

**REST** Representational State Transfer.

**RPM** Requests Per Minute.

**SPA** Single Page Application.

**TF-IDF** Term Frequency-Inverse Document Frequency.

**TPM** Tokens Per Minute.

**URL** Uniform Resource Locator.

# Noticia de ejemplo

A continuación se transcribe la noticia completa publicada por El Observador el 9 de marzo del año 2023, utilizada como ejemplo representativo de las fuentes procesadas por el sistema desarrollado.

> Dos de las principales gremiales del sector agropecuario, la Asociación Rural del Uruguay (ARU) y la Federación Rural (FR) —integrantes de Campo Unido— solicitaron al Poder Ejecutivo una reducción de la alícuota de la Contribución Inmobiliaria Rural para los propietarios de padrones rurales, considerando que persisten las dificultades generadas por la sequía.
>
> El pedido fue dirigido al presidente de la República, Luis Lacalle Pou; a la ministra de Economía y Finanzas (MEF), Azucena Arbeleche; y al ministro de Ganadería, Agricultura y Pesca (MGAP), Fernando Mattos, expresidente de la ARU.
>
> En una carta conjunta, las gremiales expusieron: "Debido a los efectos del déficit hídrico en todo el territorio nacional, solicitamos una reducción de la alícuota de la Contribución Inmobiliaria Rural a los propietarios de padrones rurales establecida por el artículo 652 de la Ley Nº 15.809, de 8 de abril de 1986".
>
> Se indicó además: "El incremento 421/022 establece que el valor real de los inmuebles para el año 2022 se determinará aplicando un coeficiente de 1,0995 a los valores reales de 2021. Considerando la tendencia a la baja del tipo de cambio nominal, el pago de la Contribución Inmobiliaria Rural en 2023 acaba siendo superior al 23 % respecto del pago del año anterior por igual concepto".
>
> ARU y FR señalaron que, si bien en años anteriores se dispusieron beneficios para determinados departamentos, la situación de sequía alcanza hoy a todo el país; por ello, solicitan que la medida se extienda a todo el territorio.
>
> La carta fue firmada por Gonzalo Valdés Requena, presidente de la ARU, y por Martín Uría Shaw, presidente de la FR. Según consultas realizadas por *El Observador*, ambos ministerios acusaron recibo del planteo; por el momento no hubo respuesta y las gremiales aguardan una definición a corto plazo.
>
> Fuente: El Observador. *"Lo que dos gremiales del agro pidieron por carta a Lacalle Pou y a dos ministros"*[1].

[1] https://www.elobservador.com.uy/nota/lo-que-dos-gremiales-del-agro-pidieron-por-carta-a-lacalle-pou-y-a-dos-ministros-202339131819

# Dominios de campos de extracción

## .1. Referencia Cartográfica

### .1.1. Dominios Geográficos

**Regiones (5 valores posibles)**

| Región | Descripción |
|---|---|
| NORESTE | Agrupamiento departamental según regionalización de UdelaR |
| SURESTE | Agrupamiento departamental según regionalización de UdelaR |
| CENTROSUR | Agrupamiento departamental según regionalización de UdelaR |
| SUROESTE | Agrupamiento departamental según regionalización de UdelaR |
| NOROESTE | Agrupamiento departamental según regionalización de UdelaR |

Tabla 1: Clasificación de Regiones

**Departamentos (19 valores posibles)**

| Departamento | Departamento |
|---|---|
| MONTEVIDEO | PAYSANDÚ |
| ARTIGAS | RÍO NEGRO |
| CANELONES | RIVERA |
| CERRO LARGO | ROCHA |
| COLONIA | SALTO |
| DURAZNO | SAN JOSÉ |
| FLORES | SORIANO |
| FLORIDA | TACUAREMBÓ |
| LAVALLEJA | TREINTA Y TRES |
| MALDONADO | |

Tabla 2: Clasificación de Departamentos

**Localidades**

- **Total:** 652 localidades según clasificación del INE[2].
- **Definición:** Áreas espaciales con presencia de viviendas agrupadas en manzanas.
- **Criterio:** Estadístico y físico según INE (2011:28).

[2] https://www.gub.uy/instituto-nacional-estadistica/datos-y-estadisticas/estadisticas/localidades-censales

# .2. Tipos de Movimientos Socioterritoriales

## .2.1. Categorización de Movimientos (11 tipos)

| Código | Tipo de Movimiento | Definición | Características |
|---|---|---|---|
| MAFC | Movimientos de la Agricultura Familiar-Campesina | Colectivos de productores/as rurales que reproducen su vida en función de la producción agropecuaria | Número de integrantes del hogar $\geq$ número de asalariados |
| MAE | Movimientos Agroecológicos | Colectivos de productores/as rurales con prácticas agroecológicas | Acción colectiva centrada en prácticas agroecológicas |
| MRA | Movimientos Ruralista-Agronegocio | Colectivos de empresas rurales | Proceso productivo basado en mano de obra asalariada |
| MTA | Movimientos de Trabajadores Agrarios | Trabajadores asalariados del sector agropecuario | Acción colectiva centrada en mejores condiciones laborales |
| MMR | Movimientos de Mujeres Rurales | Mujeres del sector rural organizadas | Acción centrada en visibilización y mejora de condiciones |
| MAM | Movimientos Ambientalistas | Colectivos enfocados en protección ambiental | Acción colectiva centrada en conservación y sustentabilidad |
| MST | Movimientos Sin Tierra | Colectivos que luchan por acceso a la tierra | Acción centrada en reforma agraria y acceso a tierra |
| MJR | Movimientos de Jóvenes Rurales | Jóvenes del sector rural organizados | Acción centrada en visibilización y mejora de condiciones |
| MINR | Movimientos Indígenas Rurales | Pueblos indígenas en territorios rurales | Acción centrada en visibilización, reconocimiento y mejora |
| MAF | Movimiento Afro | Colectivos afrodescendientes en territorios rurales | Acción centrada en visibilización, reconocimiento y mejora |
| MCA | Movimiento de Consumo Alimentario | Redes de consumo alimentario | Acción vinculada a mejora en acceso al alimento |

Tabla 3: Tipos de Movimientos Socioterritoriales

# .3. Objetivos de Acciones

## .3.1. Clasificación de Objetivos (8 tipos)

| **Objetivo** | **Definición** |
|---|---|
| Propositiva | Acción y efecto de proponer |
| Resistencia | Acción y efecto de resistir |
| Defensiva (reactiva) | Acción y efecto de defender, resistir y proteger |
| Reivindicativa | Defender a quien se halla afectado, dañado, agraviado |
| Emancipatorios (superadores) | Liberarse ante cualquier acción que exija subordinación o dependencia |
| Reprimir | Contener, detener o castigar por lo general desde el poder y con uso de violencia |
| Negociar | Tratar diversos asuntos con el fin de obtener un mejor logro |
| Cooptar | Acción original sometida a una finalidad diferente |

Tabla 4: Objetivos de Acciones de los Movimientos

# .4. Tipos de Acciones

## .4.1. Acciones Matrices (7 categorías principales)

| Acción Matriz | Definición | Criterios de Identificación |
|---|---|---|
| Manifestación/Protesta | Colectivo de personas que declaran o dan a conocer una postura pública sobre un hecho determinado | Manifestación, protesta, queja pública, movilización, reclamo colectivo |
| Vínculo con el Estado | Colectivo de personas que realizan acciones o se benefician de forma directa con la ejecución de las políticas públicas | Vínculo con el Estado, políticas públicas, interacción con organismos públicos |
| Entramados Socio-Culturales | Conjunto de relaciones sociales que se entrecruzan entre sí mediante acciones de carácter cultural | Acciones culturales comunitarias, relaciones sociales culturales |
| Comercialización | Colectivo de personas que realizan acciones vinculadas al intercambio de productos agroalimentarios | Comercialización, intercambio de productos, feria agroecológica, circuitos cortos |
| Producción | Colectivo de personas que realizan acciones vinculadas al proceso de producción de productos agroalimentarios | Producción agroalimentaria, proceso productivo, campo productivo |
| Consumo | Colectivo de personas que realizan acciones vinculadas al proceso de consumo de productos agroalimentarios | Consumo responsable, consumidores, proceso de consumo |
| Acciones con otras organizaciones | Colectivo de personas que realizan acciones de cooperación con otros colectivos | Acciones conjuntas, cooperación intermovimientos, alianzas |

Tabla 5: Acciones Matrices

## .4.2. Acciones Derivadas (por matriz)

**Manifestación/Protesta**

- **Marcha:** Desplazamiento de un grupo de personas por un espacio público
- **Paro:** Cese de las tareas cotidianas en el espacio de trabajo
- **Movilización:** Manifestación que implica la congruencia de un grupo de personas en un espacio público

- **Comunicado público:** Dar a conocer una postura colectiva mediante una nota escrita
- **Pedido de acceso a información:** Solicitud de información pública

**Vínculo con el Estado**

- **Proyectos:** Conjunto planificado de acciones en vinculación con un organismo estatal
- **Espacios de participación:** Participación en espacios de diálogo con organismos estatales
- **Integración de organismos públicos:** Integración permanente de representantes en ámbitos de gestión estatal

**Entramados Socio-Culturales**

- **Eventos:** Organización/participación en actividades para concreción de objetivos colectivos
- **Festividades:** Acción colectiva para festejar o reconocer aspectos identitarios
- **Acciones comunitarias para acceso a servicios:** Organización colectiva para acceso a servicios básicos

**Comercialización**

- **CCC (Circuitos Cortos de Comercialización):** Reducción de distancia entre producción y consumo
- **Compras públicas:** Comercialización de alimentos a organismos públicos (Ley 19292)
- **Venta conjunta:** Estrategia colectiva de comercialización
- **Compra conjunta:** Adquisición colectiva de insumos
- **Ferias agroecológicas:** Espacios de comercialización directa
- **Canastas:** Sistemas de distribución de productos

**Producción**

- **EAP (Estrategias de apoyo a la producción):** Apoyo técnico y financiero a la producción
- **Campos colectivos:** Uso compartido de tierras para producción
- **Maquinaria colectiva:** Compartir herramientas y equipos productivos

**Consumo**

- **Redes productor-consumidor:** Vínculos directos entre productores y consumidores
- **Certificación participativa:** Sistemas de garantía participativos
- **Visitas de predios:** Intercambio de experiencias productivas

# .5. Tipología de Instituciones

## .5.1. Clasificación Institucional (9 tipos)

| **Tipo** | **Descripción** | **Ejemplos** |
|---|---|---|
| Presidencia de la República/Unidades Ejecutoras | Organismos del Poder Ejecutivo | Presidencia, Unidades Ejecutoras |
| Ministerios | Ministerios del gobierno nacional | MGAP, MIDES, MEC, etc. |
| Gobiernos Departamentales | Intendencias departamentales | 19 intendencias |
| Entes autónomos/Servicios descentralizados/EPDP | Organismos autónomos y empresas públicas | UTE, ANTEL, OSE, etc. |
| Movimientos Sociales | Organizaciones de la sociedad civil | Sindicatos, cooperativas, etc. |
| Empresas privadas | Sector privado | Empresas agropecuarias, etc. |
| ONGs | Organizaciones no gubernamentales | Fundaciones, asociaciones |
| Partidos Políticos | Partidos políticos | Frente Amplio, Partido Nacional, etc. |
| Otras | Otras instituciones no clasificadas | Organismos internacionales, etc. |

Tabla 6: Tipología de Instituciones

# .6. Tipos de Acciones del Estado

## .6.1. Políticas Públicas (12 tipos)

| **Tipo** | **Definición** |
|---|---|
| Fortalecimiento institucional | Políticas que busquen mejoras en la organización, gestión y desarrollo de las organizaciones sociales |
| Asistencia técnica | Políticas cuyo objetivo sea mejorar las condiciones socioproductivas de los predios |
| Canales de comercialización | Políticas que refieren a canales comerciales ya sea para su fomento, creación o control |
| Inversiones prediales | Orientadas a la mejora productiva y/o de vida en los predios |
| Financiamiento crediticio | Propuestas de políticas respecto a créditos con múltiples destinos |
| Subsidios económicos | Subsidios económicos directos ya sea en dinero u otras formas |
| Accesos a servicios | Políticas cuyo objetivo es el fomento de servicios |
| Acciones legislativas | Acciones del Estado vinculadas directamente con el poder legislativo |
| Acciones judiciales | Acciones del Estado vinculadas con el poder judicial |
| Acceso a tierra pública | Acciones del Estado vinculadas a la política de tierras (principalmente INC) |
| Apoyo a actividades culturales | Acciones del Estado que se vinculan con la promoción de actividades culturales |
| Ámbitos de articulación | Generación de espacios interinstitucionales u organizacionales |
| Capacitación | Políticas cuyos destinatarios se busca la formación y capacitación |
| Investigación | Políticas de Ciencia, Tecnología y Sociedad |

Tabla 7: Tipos de Acciones del Estado

# .7. Objetivos de Desarrollo Sostenible (ODS)

## .7.1. Clasificación ODS (17 objetivos)

| ODS | Título | Criterios de Aplicación |
|---|---|---|
| **1** | Poner fin a la pobreza en todas sus formas | Necesidades básicas insatisfechas, acceso a servicios básicos |
| **2** | Poner fin al hambre | Agroecología, seguridad alimentaria, semillas, agricultura sostenible |
| **3** | Garantizar una vida sana y promover el bienestar | Servicios de salud pública, salud comunitaria, salud medioambiental |
| **4** | Garantizar una educación inclusiva y de calidad | Educación intercultural, educación técnica agrícola, capacitaciones |
| **5** | Lograr la igualdad entre los géneros | Derechos sexuales y reproductivos, derechos de las mujeres, acceso a propiedad |
| **6** | Garantizar la disponibilidad de agua y saneamiento | Acceso y distribución de agua, calidad para consumo humano |
| **7** | Garantizar el acceso a una energía sostenible | Expansión y asequibilidad de energías renovables |
| **8** | Promover el crecimiento económico inclusivo | Políticas de desarrollo productivo, creación de empleos decentes |
| **9** | Construir infraestructuras resilientes | Desarrollo de infraestructuras fiables y sostenibles |
| **10** | Reducir la desigualdad | Inclusión social, económica y política, igualdad de oportunidades |
| **11** | Lograr ciudades inclusivas y sostenibles | Acceso a viviendas y servicios básicos, sistemas de transporte |
| **12** | Garantizar modalidades de consumo sostenibles | Circuitos cortos, comercio justo, ferias, compras públicas |
| **13** | Adoptar medidas urgentes contra el cambio climático | Educación sobre cambio climático, mitigación y adaptación |
| **14** | Conservar y utilizar sosteniblemente los océanos | Prevención de contaminación marina, protección de ecosistemas marinos |
| **15** | Gestionar sosteniblemente los bosques | Conservación y restablecimiento de ecosistemas terrestres |
| **16** | Promover sociedades justas y pacíficas | Reducir violencia, acceso a justicia |
| **17** | Revitalizar la Alianza Mundial para el Desarrollo | Apoyo internacional en finanzas, tecnologías, comercio |

Tabla 8: Objetivos de Desarrollo Sostenible

## .7.2. Fuentes de Información

- **Manual base:** Manual Metodológico del Proyecto Movimientos Socioterritoriales Rurales en Perspectiva Comparada para el Caso del Uruguay
- **Organismo:** OCAU (Observatorio de la Cuestión Agraria en Uruguay)
- **Fecha:** Junio 2025
- **Instituciones:** UDELAR, UNESP, CLACSO