



Universidad Austral de Chile

Facultad de Ciencias de la Ingeniería
Escuela de Ingeniería Civil en Informática

DESARROLLO DE UNA ARQUITECTURA DE SOFTWARE PARA LA GENERACIÓN AUTOMATIZADA DE REPORTE

Proyecto para optar al título de
Ingeniero Civil en Informática

PROFESOR PATROCINANTE:
MATTHIEU VERNIER
DOCTOR EN INFORMÁTICA

CO-PATROCINANTE
FABIAN ALEXIS RUIZ FLORES
INGENIERO CIVIL EN INFORMÁTICA
MAGÍSTER EN CIENCIA DE LOS DATOS

PROFESOR INFORMANTE
CRISTIAN OLIVARES-RODRÍGUEZ
DOCTOR EN INGENIERÍA

MARCELO IGNACIO SOTO BARRÍA

VALDIVIA – CHILE
2023

ÍNDICE

ÍNDICE	I
ÍNDICE DE TABLA	III
ÍNDICE DE FIGURA	IV
RESUMEN	VI
ABSTRACT	VII
1. INTRODUCCIÓN	1
1.1 Antecedentes sobre el equipo Sophia	1
1.2 Antecedentes sobre la tecnología desarrollada entre 2019 y 2022 por el equipo Sophia	2
1.2.1 Sophia: una tecnología para recopilar, indexar y analizar datos de medios de prensa	2
1.3 Una primera oportunidad detectada al inicio del año 2022: monitorear los medios de prensa para el seguimiento de indicadores territoriales	3
1.4 Problemática general: ¿Cómo desarrollar una arquitectura que permita automatizar la generación de reportes?	3
1.4.1 Carencias identificadas en la arquitectura previa de Sophia	5
1.5 Problemática específica: ¿Cómo mejorar la arquitectura actual para: facilitar su evolución y mantenibilidad?	5
1.5.1 Concientizar la importancia de la calidad de software mediante criterios, refactorización y prevención de hediondez de código	5
1.6 OBJETIVOS DEL PROYECTO	6
1.6.1 Objetivo General	6
1.6.2 Objetivos Específicos	6
2. MARCO TEÓRICO	7
2.1 ¿Cuáles son los criterios de calidad de software?	7
2.2 ¿Cuáles son los diagramas que se pueden utilizar para modelar una arquitectura de software?	7
2.2.1 Modelo de Vistas de Arquitectura 4+1	8
2.2.2 Diagrama Caso de uso	8
2.2.3 Diagrama Entidad Relación	10
2.2.4 Diagrama Modelo Relacional	10
2.3 ¿Cuáles son las herramientas para desplegar arquitecturas y realizar pruebas?	10
2.3.1 Protocolo CI/CD	11
2.4 Ejemplos de reportes PDF de percepción mediática para instituciones públicas: los Policy Brief y el Clipping de Medios	11
2.4.1 Policy Brief	12
2.4.2 Clipping de medios	12
2.4.3 Integración e impacto de los reportes	12
3. DIAGNÓSTICO DE LA ARQUITECTURA PREVIA DE SOPHIA	13
3.1 ¿Quiénes son los usuarios de Sophia versión 1 y cómo utilizan el software?	13
3.2 ¿Cómo se implementan los casos de uso en Sophia versión 1?	14
3.3 Caso de uso: lanzar la recopilación de nuevas noticias	15
3.4 Caso de uso: añadir nuevos códigos de scrapers de medios de prensa	19
3.5 Caso de uso: extraer un dataset de noticias de prensa a partir de keywords y fecha	22
4. DIAGNÓSTICO DE LOS COMPONENTES DE SOPHIA	24
4.1 Ground-control: Centro de pilotaje de Sophia	24
4.1.1 ¿Cómo se despliega Sophia versión 1?	24
4.2 Caleuche: Recopilación de noticias de prensa	25
4.2.1 Base de datos Caleuche: Modelo Entidad-Relación	26
4.2.2 Base de datos Caleuche: Modelo Relacional	26
4.3 Sun: Almacenamiento de las noticias de prensa	27
4.3.1 Base de datos Sun: Modelo Entidad-Relación	28
4.3.2 Base de datos Sun: Modelo Relacional	28
4.4 Sun-search: Búsqueda en las noticias de prensa	30

4.4.1 Motor de búsqueda en documentos: Sun-search	30
5. DISCUSIÓN SOBRE EL DIAGNÓSTICO REALIZADO	31
5.1 Identificación de problemas (malas prácticas)	31
5.1.1 Falta de proceso formal de diseño arquitectónico	31
5.1.2 Falta de proceso formal de integración y despliegue continuos	31
5.1.3 Falta de manuales para los usuarios de Sophia.....	32
5.1.4 Desorganización en los archivos	32
5.1.5 Problemas en el código (código redundante, credenciales en el código) (Relacionado con la presencia de código redundante y la aparición de credenciales en ciertos archivos)	32
5.2 Recomendaciones y refactorización prioritaria	33
6. DISEÑO E IMPLEMENTACIÓN DE COMPONENTES DE SOFTWARE PARA LA GENERACIÓN DE REPORTES PDF DE PERCEPCIÓN MEDIÁTICA	34
6.1 Análisis de requisitos y nuevo caso de uso	34
6.2 Decisiones previas a la implementación de la generación de reportes	35
6.3 Evaluación y elección de Herramientas para Generación Automática	36
6.4 Implementación de Herramienta para Generación Automática	37
6.4.1 Implementar Caso de Uso: “Extraer un Dataset”	37
6.4.2 Implementar Caso de Uso: “Generar un reporte”	44
6.5 Implementación de los resultados del Análisis.....	47
7. DISCUSIÓN Y CONCLUSIÓN	48
8. REFERENCIAS	50
9. ANEXOS	51

ÍNDICE DE TABLA

TABLA	PÁGINA
Tabla 1: Atributos de calidad validados por el modelo de calidad ISO25010.....	7

ÍNDICE DE FIGURA

FIGURA	PÁGINA
Figura 1: Diagrama de casos de uso de Sophia versión 1.....	4
Figura 2: Diagrama de casos de uso de Sophia versión 2.....	4
Figura 3: Tabla con los respectivos símbolos y sus funciones de un diagrama de flujo....	9
Figura 4: Línea de tiempo de versiones de Sophia.	13
Figura 5: Diagrama de casos de uso de Sophia versión 1.....	14
Figura 6: Diagrama de flujo para la recopilación de nuevas noticias en Sophia versión 1.	16
Figura 7: (Continuación).....	16
Figura 8: (Continuación).....	17
Figura 9: (Continuación).....	18
Figura 10: (Continuación).....	19
Figura 11: Diagrama de flujo para añadir nuevos códigos de scrapers de medios de prensa en Sophia versión 1.....	20
Figura 12: (Continuación).....	21
Figura 13: (Continuación).....	21
Figura 14: Código Python para la extracción de un dataset.....	22
Figura 15: (Continuación).....	22
Figura 16: (Continuación).....	23
Figura 17: (Continuación).....	23
Figura 18: Componente “ground-control” de la versión 1 de Sophia.....	24
Figura 19: Arquitectura completa de Sophia con sus respectivos componentes.	25
Figura 20: Componente "Caleuche" de la versión 1 de Sophia.	25
Figura 21: Modelo Entidad-Relación de la base de datos Caleuche.....	26
Figura 22: Modelo Relacional de la base de datos Caleuche.....	27
Figura 23: Componente "sun" de la versión 1 de Sophia.....	28
Figura 24: Modelo Entidad-Relación de la base de datos Sun	28
Figura 25: Modelo Relacional de la base de datos Sun	29
Figura 26: Componente "sun-search" de la versión 1 de Sophia.....	30
Figura 27: Archivos obsoletos en Ground -Control.....	32
Figura 28: Código redundante dentro de la arquitectura de Sophia.....	33
Figura 29: Archivos de Componente Caleuche de producción.....	33
Figura 30: Diagrama de casos de uso de Sophia versión 2.....	35
Figura 31: Código de importación de librerías y paquetes necesarios.....	38
Figura 32: Código de los parámetros para la búsqueda en Sophia.	39
Figura 33: Widget de selección de región.....	39
Figura 34: Widget de selección de medios de prensa.	39
Figura 35: Widget de selección de fechas (inicio y termino).....	40
Figura 36: Código de búsqueda de noticias dentro de Elasticsearch.	41
Figura 37: Código de guardado de noticias encontradas en un Dataset.....	41
Figura 38: Código de limpieza del dataset.....	42
Figura 39: Código para asignar y ordenar el dataset dada las palabras claves.	42
Figura 40: Código para etiquetar las comunas mencionadas.....	43

Figura 41: Código para asignar porcentaje de sentimiento a las noticias.	43
Figura 42: Código para guardar variables en un archivo JSON.....	44
Figura 43: Código de importación de librerías y paquetes necesarios.....	44
Figura 44: Código de importación de librerías y paquetes necesarios.....	45
Figura 45: Código para generación de mapas de calor geográfico.	45
Figura 46: Mapa de calor geográfico respecto al análisis de sentimientos.	46
Figura 47: Gráfico de relación entre número de habitante y cobertura mediática sobre Turismo.	46
Figura 48: Mapa de calor respecto a la importancia sobre la temática, comunas y mes.	47

RESUMEN

El presente trabajo, asociado al proyecto Fondecyt de Iniciación n° 11190714, "Sophia2: métodos basados en lingüística computacional y machine learning para analizar el pluralismo en los medios de prensa", llevado a cabo en la Facultad de Ciencias de la Ingeniería (FCI) de la Universidad Austral de Chile desde diciembre de 2019 hasta marzo de 2023, tuvo como objetivo central mejorar la arquitectura de extracción de noticias y el proceso de generación de reportes.

Para llevar a cabo la mejora de la arquitectura inicial, se implementó un proceso de reingeniería la cual surgió debido a la ausencia de un flujo coherente en su desarrollo. En este contexto, se llevaron a cabo pasos fundamentales, incluyendo la creación de diagramas de flujo para visualizar de manera clara el proceso, la identificación de casos de uso esenciales y la definición de procedimientos eficientes. Por último, se llevó a cabo un análisis detallado del código existente, resaltando puntos clave que podrían ser objeto de mejoras en el futuro.

Posteriormente, se enfocó en la generación de reportes, priorizando el trabajo del analista. Se investigó y desarrolló de manera óptima, adaptable y escalable la generación de reportes, optando por utilizar el software Jupyter Notebook. Dos notebooks fueron creados: el primero para definir parámetros de extracción de noticias y el segundo para generar gráficos con diversas librerías, proporcionando herramientas visuales para implementar en reportes futuros.

En la fase final, se realizó un análisis detallado del trabajo, discutiendo futuras implementaciones y mejoras para la arquitectura general de Sophia.

ABSTRACT

The present work, associated to the Fondecyt de Iniciación project n° 11190714, "Sophia2: methods based on computational linguistics and machine learning to analyze pluralism in the press media", carried out at the Faculty of Engineering Sciences (FCI) of the Universidad Austral de Chile from December 2019 to March 2023, had as central objective to improve the news extraction architecture and the report generation process.

To carry out the improvement of the initial architecture, a reengineering process was implemented, which arose due to the absence of a coherent flow in its development. In this context, fundamental steps were carried out, including the creation of flow charts to clearly visualize the process, the identification of essential use cases and the definition of efficient procedures. Finally, a detailed analysis of the existing code was carried out, highlighting key points that could be improved in the future.

Subsequently, we focused on report generation, prioritizing the analyst's work. Optimal, adaptable and scalable reporting was investigated and developed, opting to use the Jupyter Notebook software. Two notebooks were created: the first to define news extraction parameters and the second to generate graphics with various libraries, providing visual tools to implement in future reports.

In the final phase, a detailed analysis of the work was performed, discussing future implementations and improvements to the overall Sophia architecture.

1. INTRODUCCIÓN

1.1 Antecedentes sobre el equipo Sophia

Este proyecto de título se inscribe desde septiembre de 2022 en las actividades del equipo Sophia donde colaboran investigadores, ingenieros y estudiantes de la Facultad de Ciencias de la Ingeniería y de la Facultad de Filosofía y Humanidades de la Universidad Austral de Chile.

Desde 2016, Sophia lidera y participa en distintos proyectos de investigación vinculados a las ciencias sociales computacionales, al análisis de los medios de comunicación, a la ingeniería de datos y al tratamiento automático del lenguaje. Su actividad se inicia principalmente a partir de tres proyectos financiados por ANID (Cárcamo, L., Scheihing, E., Vernier, M., Aguilera, O. (2015-2019), Vernier, M. (2019-2023) y Cárcamo, L., Cárdena, C., Vernier, M., Scheihing, E., Sáez, D. (2021-2022)) en temas de análisis de medios de comunicación. A la fecha, estos proyectos permitieron concretar siete publicaciones científicas, diez trabajos de título de pregrado en Informática o Periodismo, dos tesis de Magíster en Informática o Comunicación. Además, dos tesis de Doctorado en Comunicación están en curso.

Además de su actividad de investigación, el equipo Sophia participa en proyectos de vinculación con el medio socio productivo. Entre 2021 y 2023, implementa y mantiene la arquitectura de software de FuSA, un sistema inteligente para el reconocimiento de fuentes sonoras en archivos audios desarrollado por la Facultad de Ciencias de la Ingeniería, en colaboración con el Ministerio de Medio Ambiente, y empresas tecnológicas en acústica como VitGlobal, ControlAcústica, Capta y Acústica Marina en el marco de un proyecto ANID FONDEF Suárez, E., Vernier, M., Huijse, P., Arenas, J., Poblete, V. (2020-2023). En 2017-2018, a través de un proyecto CORFO (Vernier, M. (2017)) trabajó con la empresa chilena Octopull para mejorar un sistema de chat inteligente. En 2022, a través de contratos I+D, desarrolló modelos y métodos de extracción y clasificación de documentos PDF para la empresa FinTree. En 2021-2022, colaboró con un equipo de la Pontificia Universidad Católica para proporcionar datos de medios de prensa regionales en la plataforma de gestión estratégica del Gobierno Regional de Los Lagos. Finalmente, desde 2023, Sophia trabaja con el Laboratorio Natural Andes del Sur en el desarrollo de la plataforma Conversa con Los Andes. Detallamos estos dos últimos proyectos en la sección 1.3 ya que motivan este proyecto de título.

Finalmente, desde 2023, el equipo Sophia se encuentra ejecutando un proyecto Corfo Semilla Inicia, en el cual se formalizó la empresa *Sophia Language Technologies* SpA. Esta empresa se encuentra en una fase inicial de definición de su modelo de negocio y validación comercial. Esta empresa busca a corto plazo posicionar primeras ofertas en los mercados del monitoreo de indicadores territoriales y del monitoreo de medios de prensa. La visión estratégica de la empresa sigue las tendencias del mercado global de los *Smart*

Territories (Kiran, 2022) y los desafíos actuales para la construcción de la democracia (Sabot, 2023).

1.2 Antecedentes sobre la tecnología desarrollada entre 2019 y 2022 por el equipo Sophia

1.2.1 Sophia: una tecnología para recopilar, indexar y analizar datos de medios de prensa

A lo largo de sus distintos proyectos, el equipo Sophia desarrolló una tecnología de recopilación, almacenamiento y análisis de documentos textuales, en particular de noticias publicadas por medios de prensa chilenos e internacionales. En un inicio, se trataba de una herramienta de uso interno para facilitar la ejecución de proyectos de investigación en ciencias sociales computacionales. En particular, un importante desarrollo se realizó para responder a las necesidades de la tesis de Magíster en Informática de Fabian Ruiz (2020) titulada: "Evolution of the gender biases in the news media: a computational method using dynamic topic model" (Ruiz, 2020). En este sentido, la tecnología se desarrolló sin seguir un proceso de ingeniería de software riguroso, y carece históricamente de ciertas debilidades en aspectos de modelamiento y documentación. Sin embargo, el software Sophia se desarrolló haciendo un particular énfasis en los criterios de calidad de desplegabilidad y escalabilidad. Utiliza tecnologías de Cloud Computing, automatización de despliegue como Docker y Terraform, protocolos de comunicación como RabbitMQ y de seguridad como Vault, entre otros. En el capítulo 3, se presenta un diagnóstico detallado de la arquitectura de Sophia en su versión previa al presente proyecto. Sus principales desarrolladores son Fabian Ruiz y Matthieu Vernier, miembros permanentes del equipo Sophia, y varios estudiantes en informática que contribuyeron puntualmente a través de su trabajo de título (en particular Eleazar Vásquez (Vásquez, 2022), Nicolas Treimun, Felipe Saldías (Saldías, 2021), Natalia Ramírez).

En resumen, la tecnología Sophia consta de dos componentes principales de software: (1) uno dedicado a la recopilación masiva de noticias de medios de prensa a través de técnicas de scraping y a su almacenamiento en bases de datos, (2) y el otro dedicado a la extracción de datasets para poder analizar los datos desde scripts Python. Los usuarios del sistema, investigadores en ciencias sociales computacionales, pueden extraer datasets a través de un motor de búsqueda disponible en la URL: <http://search.sophia2.org/>.

Actualmente, Sophia indexa datos de medios de prensa de 41 países. Por ejemplo, se trata de aproximadamente tres millones de noticias de prensa indexada para Chile, 750.000 para España, etc. Los datos indexados tienen el formato siguiente: id, fecha de publicación, nombre del medio de prensa, país, título, contenido textual de la noticia, url de la noticia. A la fecha, cinco investigadores utilizan frecuentemente Sophia para su investigación o actividades de docencia.

1.3 Una primera oportunidad detectada al inicio del año 2022: monitorear los medios de prensa para el seguimiento de indicadores territoriales

Entre 2021 y 2022, el equipo Sophia participó en el proyecto "Estrategia Regional de Desarrollo Lagos 2030", liderado por el Gobierno Regional de Los Lagos y un equipo de la Pontificia Universidad Católica. En este proyecto, se utilizó la arquitectura de Sophia para monitorear las noticias publicadas en 25 medios de comunicación regionales de Los Lagos y se integró con una plataforma de seguimiento de indicadores georreferenciados del Gobierno Regional de Los Lagos. Por otra parte, el equipo Sophia mostró un primer prototipo de diagnóstico de las problemáticas territoriales basado en el análisis automatizado de noticias de prensa. Este primer prototipo consistió en un reporte en PDF construido manualmente que describe el territorio de Los Lagos en relación con dos problemáticas territoriales ("cambio climático" y "tráfico de drogas") e indicadores que resumen la percepción mediática.

En esta investigación se ha detectado la oportunidad de ofrecer un sistema de monitoreo de medios de prensa especializado en el seguimiento de problemáticas territoriales. Tal como mencionado por miembros del equipo del GORE Los Lagos, estos reportes de monitoreo permitirían abordar contingencias a corto plazo y orientar la Estrategia Regional de Desarrollo a mediano y largo plazo. Además, se observa la creciente inversión de instituciones públicas y privadas en soluciones de monitoreo de medios de prensa. Esto se ha evidenciado a través de la revisión de portales web especializados en licitaciones públicas, como "todolicitaciones.cl", donde se ha detectado la contratación de servicios de monitoreo y análisis de medios por parte de diversas instituciones. Un ejemplo de esta tendencia es la licitación de la municipalidad de Providencia por un monto de \$16.500.000 adjudicada a la empresa Conecta Research.

1.4 Problemática general: ¿Cómo desarrollar una arquitectura que permita automatizar la generación de reportes?

Para responder a la oportunidad detectada, es necesario adaptar la arquitectura actual de Sophia, la cual no fue diseñada para generar reportes automatizados. La Figura 1 muestra los casos de uso de la arquitectura Sophia versión 1 (31/12/2022) y la Figura 2 muestra los casos de uso de la arquitectura Sophia versión 2 imaginando funcionalidades para automatizar la generación de reportes. En los capítulos 3 y 4, se detalla un análisis de estos diagramas. Esta adaptación de lo existente requiere diseñar e implementar nuevos componentes y analizar cómo deberían interactuar con los componentes existentes.

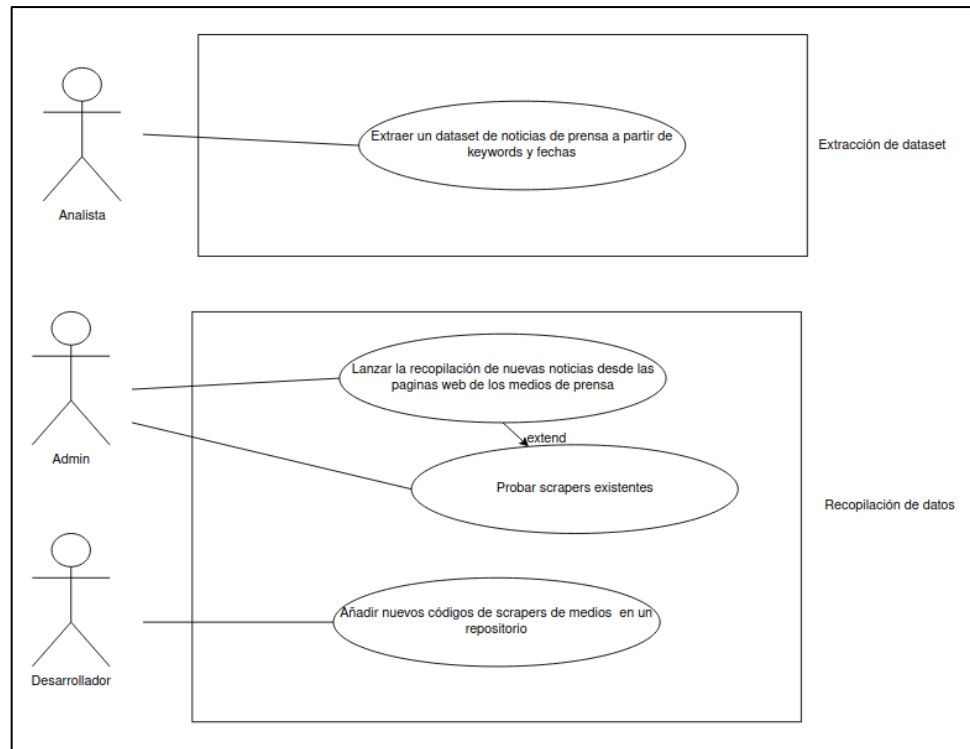


Figura 1: Diagrama de casos de uso de Sophia versión 1.

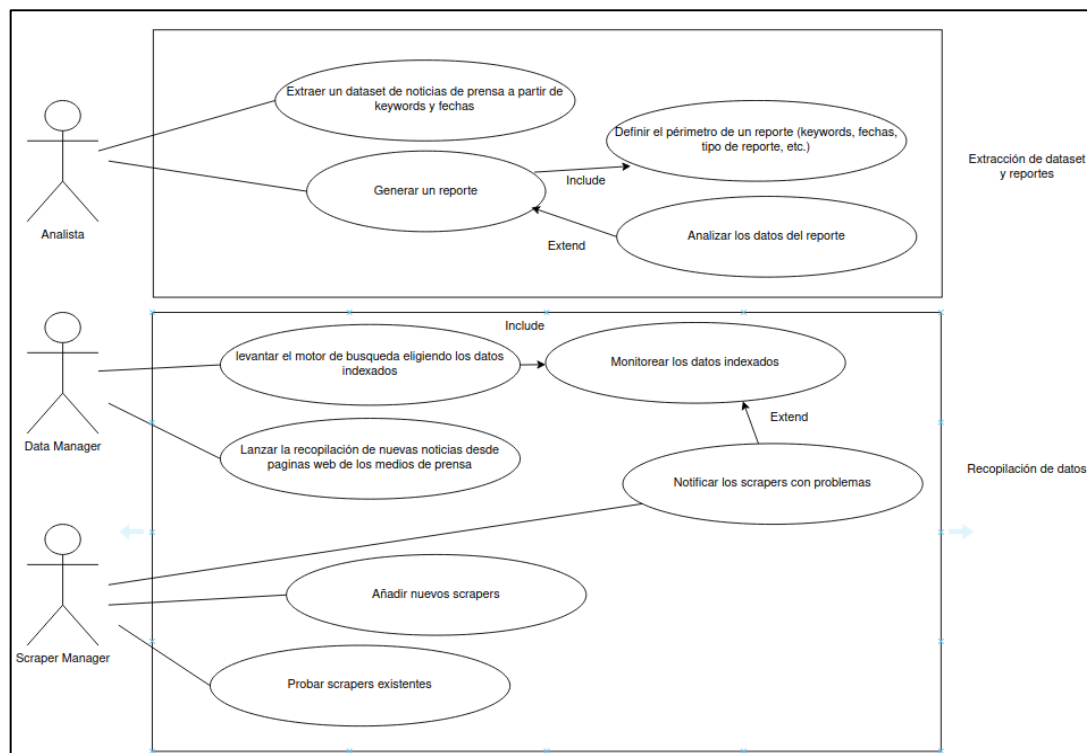


Figura 2: Diagrama de casos de uso de Sophia versión 2.

1.4.1 Carencias identificadas en la arquitectura previa de Sophia

Aunque el software Sophia ya está en producción y responde a necesidades de ciertos usuarios, su arquitectura actual carece de documentación y diagramas que permitan su mantenimiento y evolución de manera efectiva. Estas carencias complican la incorporación de nuevos componentes. Por lo tanto, este proyecto implica la necesidad de describir y documentar la arquitectura previa y nueva a través de un proceso de diseño riguroso, mejorando la calidad del software y planteando la adaptabilidad como requisito no funcional prioritario.

1.5 Problemática específica: ¿Cómo mejorar la arquitectura actual para: facilitar su evolución y mantenibilidad?

1.5.1 Concientizar la importancia de la calidad de software mediante criterios, refactorización y prevención de hediondez de código

La descripción y documentación de la arquitectura del software en el proyecto Sophia tiene como objetivo fundamental mejorar la calidad y garantizar la adaptabilidad del sistema. En este contexto, la calidad del software es crucial para el éxito y la eficiencia, por lo tanto, al asegurar una alta calidad en el software se cumplen con los requisitos funcionales y no funcionales necesarios para satisfacer las necesidades de los usuarios.

Los criterios de calidad que se aplicarán incluyen la fiabilidad, la mantenibilidad y la portabilidad, entre otros. Estos criterios son de suma importancia para el proyecto, ya que Sophia se basa en la evolución y mejora constante del software existente, así como en la incorporación de nuevos componentes. La calidad del software garantiza que Sophia sea confiable, fácil de mantener y capaz de adaptarse a los cambios futuros que puedan surgir.

Es crucial prestar atención a la detección y prevención de problemas de hediondez de código, también conocidos como “code smells”. La refactorización del código desempeña un papel fundamental en Sophia, ya que ayuda a mantener un código limpio, legible y fácil de mantener a largo plazo. Evitar la acumulación de malas prácticas de codificación es esencial para asegurar que Sophia siga siendo un software robusto y de alta calidad, capaz de adaptarse a las demandas cambiantes de los usuarios.

1.6 OBJETIVOS DEL PROYECTO

1.6.1 Objetivo General

Desarrollar una arquitectura de software que permite generar reportes PDF de percepción mediática tomando decisiones que apuntan a satisfacer requisitos no funcionales prioritarios.

1.6.2 Objetivos Específicos

1. Realizar un diagnóstico de la arquitectura inicial del software Sophia.
2. Diseñar componente(s) de software necesario(s) para permitir la generación de reportes PDF de percepción mediática.
3. Implementar componente(s) de Software e integrar en arquitectura de Sophia.
4. Desplegar la arquitectura en un ambiente de producción y realizar pruebas.

2. MARCO TEÓRICO

2.1 ¿Cuáles son los criterios de calidad de software?

La calidad en el diseño de software es esencial para cumplir con las especificaciones y requisitos, tanto funcionales como no funcionales. Existen diversas metodologías y prácticas recomendadas para mejorar la calidad del software, incluyendo el enfoque en la calidad funcional y estructural. La calidad funcional se refiere a la capacidad del software para cumplir con los requisitos funcionales mientras que la calidad estructural puede incluir el rendimiento, la escalabilidad, la mantenibilidad y la seguridad. Estas características son reconocidas como atributos de calidad (ver Tabla 1).

Para medir estos criterios se utilizan técnicas de evaluación como la revisión de código y el análisis de casos de uso. Además, la aplicación de buenas prácticas y metodologías, como las metodologías ágiles y el desarrollo guiado por pruebas, es fundamental para garantizar un diseño de software de alta calidad

Tabla 1: Atributos de calidad validados por el modelo de calidad ISO25010.

Adecuación Funcional	Eficiencia de Desempeño	Compatibilidad	Usabilidad	Fiabilidad	Seguridad	Mantenibilidad	Portabilidad
Compleitud funcional Corrección funcional Pertinencia funcional	Comportamiento temporal Utilización de recursos Capacidad	Coexistencia Interoperabilidad	Inteligibilidad Aprendizaje Operabilidad Protección frente a errores de usuario Estética Accesibilidad	Madurez Disponibilidad Tolerancia a fallos Capacidad de recuperación	Confidencialidad Integridad No repudio Autenticidad Responsabilidad	Modularidad Reusabilidad Analizabilidad Capacidad de ser modificado Capacidad de ser probado	Adaptabilidad Facilidad de instalación Capacidad de ser reemplazado

2.2 ¿Cuáles son los diagramas que se pueden utilizar para modelar una arquitectura de software?

Para el diseño y desarrollo de arquitecturas de software eficientes es fundamental contar con herramientas que permitan modelar de manera precisa y completa la estructura del sistema en cuestión. En este sentido, los diagramas son una herramienta indispensable para representar visualmente los diferentes componentes de un sistema de software y las relaciones entre ellos.

2.2.1 Modelo de Vistas de Arquitectura 4+1

El Modelo de Vistas de Arquitectura 4+1 es un enfoque para el diseño y documentación de sistemas de software propuesto por Philippe Kruchten. Esta metodología utiliza cinco tipos de diagramas para describir diferentes aspectos de una arquitectura de software.

Los cinco tipos de diagramas del Modelo de Vistas de Arquitectura 4+1 son los siguientes:

1. Vista de escenarios (Scenarios View): Esta vista se centra en describir los diferentes escenarios de uso del sistema. Utiliza diagramas de casos de uso y diagramas de secuencia para representar las interacciones entre los actores y el sistema en diferentes situaciones.
2. Vista de desarrollo (Development View): En esta vista, se enfoca en la estructura del sistema y cómo se organiza su código fuente. Los diagramas de componentes y diagramas de despliegue se utilizan para representar los módulos del sistema y las dependencias entre ellos.
3. Vista lógica (Logical View): Aquí se describe la estructura interna del sistema y cómo se organizan sus componentes y clases. Los diagramas de clases y diagramas de paquetes se utilizan para representar las relaciones y la composición de las entidades lógicas del sistema.
4. Vista de procesos (Process View): Esta vista se centra en la ejecución del sistema y cómo se distribuyen las tareas y los recursos en un entorno de ejecución. Los diagramas de actividad y diagramas de hilos se utilizan para representar los flujos de trabajo y las interacciones concurrentes dentro del sistema.
- +1. Vista de escenarios (Scenarios View): Esta vista adicional describe casos de uso concretos que ejemplifican y validan la arquitectura propuesta. Los diagramas de escenarios arquitectónicos se utilizan para representar situaciones específicas de uso del sistema.

La importancia de aplicar el Modelo de Vistas de Arquitectura 4+1 en una arquitectura de software es relevante en el diseño de arquitecturas de software porque brinda claridad, facilita la comunicación, promueve un diseño bien estructurado, proporciona documentación clara y ayuda en la validación y verificación de la arquitectura propuesta.

2.2.2 Diagrama Caso de uso

Un aspecto importante en el desarrollo de software es la comprensión y documentación de los requisitos funcionales del sistema. Una técnica utilizada para lograr esto es el uso de esta herramienta. Un caso de uso es una descripción detallada de los pasos o actividades que deben realizarse para llevar a cabo algún proceso en el sistema. Estos describen cómo un actor interactúa con el sistema para lograr un objetivo específico. Un actor puede ser un usuario humano, otro sistema o un proceso.

Estos suelen representarse gráficamente mediante diagramas. Estos diagramas muestran los actores, los casos de uso y las relaciones entre ellos. La creación de un diagrama implica la identificación de los actores y los casos de uso, y el establecimiento de las relaciones entre ellos. Esto ayuda a definir el alcance del sistema y a asegurar que todos los requisitos funcionales sean considerados.

La creación de diagramas es una tarea prioritaria en el desarrollo de software porque ayuda a asegurar que todos los requisitos funcionales sean considerados y documentados. Además, estos diagramas pueden ser utilizados como una herramienta de comunicación entre los desarrolladores y los stakeholders para asegurar que todos comprendan y estén de acuerdo con los requisitos del sistema.

2.2.2 Diagrama de flujo

Los diagramas de flujo son especialmente útiles para representar procesos que involucran decisiones y ramificaciones. Al utilizar diferentes tipos de cajas para representar diferentes tipos de pasos, como operaciones, decisiones y entradas/salidas, se puede visualizar claramente el flujo de la ejecución y cómo las decisiones afectan el resultado del proceso.

En la Figura 3 se definirán los símbolos utilizados en el diagrama de flujo con sus respectivas definiciones. Esto ayudará a entender mejor el significado de cada símbolo y cómo se utilizan para representar los diferentes tipos de pasos en el proceso.






Símbolo		Función
Líneas de flujo		Conectan los pasos, etapas, decisiones y otros elementos que intervienen en los diagramas
Decisión		Se usan para indicar las elecciones y decisiones realizadas.
Datos		Ofrecen información nueva, de interés o de gran valor para el desarrollo del proceso representado.
Actividad		Indican las acciones que se transforman en datos que dan continuidad al proceso.
Inicio / final		Se utiliza cada vez que se indica el problema/ solución en el diagrama de flujo marcando el inicio y cierre de mismo.

Figura 3: Tabla con los respectivos símbolos y sus funciones de un diagrama de flujo.

En el contexto del desarrollo de software, los diagramas de flujo pueden ser utilizados para representar el flujo de ejecución de un programa o algoritmo. Al visualizar el flujo de la ejecución en un diagrama de flujo, se puede entender mejor cómo funciona el código

y cómo interactúan las diferentes partes del sistema. Esto puede ayudar a identificar posibles problemas o áreas de mejora en el código.

2.2.3 Diagrama Entidad Relación

Un diagrama de entidad-relación (ERD) es un tipo de diagrama de flujo que ilustra cómo las entidades (personas, objetos o conceptos) se relacionan entre sí dentro de un sistema. Los diagramas ER se utilizan con mayor frecuencia para diseñar o depurar bases de datos relacionales en campos como la ingeniería de software, los sistemas de información empresarial, la educación y la investigación.

En la arquitectura de Sophia, no existe un diagrama de entidad-relación. Esto significa que no se ha creado un diagrama visual que muestre cómo las diferentes entidades en el sistema se relacionan entre sí. Esto no significa que no existan relaciones entre las entidades en el sistema; significa que estas relaciones no se han representado visualmente en un diagrama ER.

2.2.4 Diagrama Modelo Relacional

El modelo relacional es un enfoque para organizar y representar datos en una base de datos utilizando tablas, filas y columnas. Cada tabla representa una entidad del mundo real, y las relaciones entre las entidades se establecen mediante claves primarias y claves externas. Estas relaciones permiten consultar y manipular los datos de manera eficiente, manteniendo la integridad referencial.

En la arquitectura de Sophia, se sigue el modelo relacional de bases de datos, pero no se utiliza un diagrama de entidad-relación (ERD) para visualizar las relaciones entre las entidades. Aunque existen relaciones entre las entidades en Sophia, estas no se representan gráficamente en un diagrama ER. La arquitectura se enfoca en organizar y manipular los datos eficientemente, basándose en el modelo relacional, pero prescindiendo de un diagrama ER para mostrar las relaciones.

2.3 ¿Cuáles son las herramientas para desplegar arquitecturas y realizar pruebas?

El despliegue y la gestión de una arquitectura de software moderna no solo se trata de automatizar y administrar la infraestructura necesaria, sino también de garantizar la calidad del software y realizar pruebas eficientemente. En este sentido, herramientas como Terraform y Docker juegan un papel fundamental.

Terraform es una herramienta de "infraestructura como código" que permite orquestar el despliegue de la arquitectura deseada de forma declarativa y versionada. Esto significa que se puede describir la infraestructura en un archivo de configuración, lo que facilita la replicación de entornos de desarrollo, pruebas y producción. Al establecer una infraestructura consistente en todos los entornos, se reducen los errores y se mejora la

calidad del software. Además, Terraform permite realizar cambios y actualizaciones de manera controlada y reversible, lo que contribuye a la estabilidad y confiabilidad del sistema.

Por otro lado, Docker es una tecnología de contenedores que proporciona un entorno aislado y reproducible para las aplicaciones y sus dependencias. Esto facilita la creación de entornos de pruebas consistentes, donde se pueden ejecutar las pruebas de manera confiable y repetible. Los contenedores Docker también permiten simular diferentes configuraciones y escenarios, lo que mejora la cobertura de pruebas y la detección temprana de posibles problemas.

En conjunto, Terraform y Docker no solo brindan automatización y eficiencia en el despliegue de arquitecturas de software, sino que también contribuyen a mejorar la calidad del software. Estas herramientas permiten establecer entornos consistentes, realizar cambios controlados, simplificar la replicación de entornos de pruebas y aumentar la cobertura de pruebas. Todo esto resulta en un desarrollo más confiable y robusto de aplicaciones.

2.3.1 Protocolo CI/CD

El protocolo de CI/CD (Integración Continua/Despliegue Continuo) es una metodología y conjunto de prácticas que se utilizan en el desarrollo de software para mejorar la eficiencia y la calidad en la entrega de aplicaciones. Consiste en la Integración Continua, que implica la integración regular del código desarrollado por diferentes miembros del equipo en un repositorio compartido, junto con pruebas automáticas para detectar errores tempranamente. Además, incluye el Despliegue Continuo, que automatiza la entrega y despliegue de la aplicación en diferentes entornos, reduciendo los tiempos de espera y los posibles errores humanos.

La implementación del protocolo de CI/CD en el proyecto Sophia implica la automatización de procesos, lo que mejora la calidad del software y acelera el desarrollo y despliegue de nuevas funcionalidades. Al detectar errores tempranamente, se garantiza la estabilidad del sistema y se facilita su adaptabilidad. Esto contribuye a una evolución más eficiente y a una mayor facilidad para realizar tareas de mantenimiento en la arquitectura del proyecto Sophia.

2.4 Ejemplos de reportes PDF de percepción mediática para instituciones públicas: los Policy Brief y el Clipping de Medios

Se presentarán dos tipos de reportes en formato PDF que las instituciones públicas suelen utilizar para analizar la percepción mediática sobre una problemática específica. Estos reportes, conocidos como Policy Brief y Clipping de medios, son herramientas importantes para resumir y comprender la información mediática relevante en el ámbito de las políticas públicas. Además, se explorará cómo la tecnología Sophia puede desempeñar un papel clave en el análisis y la generación de estos reportes.

2.4.1 Policy Brief

El Policy Brief constituye un documento esencial en el ámbito de la investigación y análisis de políticas públicas, con el propósito de suministrar información pertinente y valiosa a los responsables de tomar decisiones y otros actores fundamentales. Este tipo de informe se centra en el análisis y resumen de una política pública específica o de un problema vinculado a una política pública particular. Se abordará su estructura y contenido, que típicamente comprende una introducción al tema, una descripción de la política pública y su contexto, un análisis de las implicaciones y recomendaciones para mejorar la política pública.

2.4.2 Clipping de medios

El Clipping de medios es otro tipo de reporte utilizado por las instituciones públicas para analizar la percepción mediática sobre una problemática específica. Consiste en recopilar y resumir noticias y artículos de medios de comunicación relevantes para la institución. Se explicará su función y cómo se recopilan los datos de los medios de prensa utilizando la tecnología Sophia. Además, se discutirán los beneficios de utilizar el Clipping de medios, como monitorear la cobertura mediática, identificar tendencias y ajustar las estrategias de comunicación en consecuencia.

2.4.3 Integración e impacto de los reportes

Se explorará cómo la tecnología Sophia puede desempeñar un papel clave en el análisis y la generación de los reportes de Policy Brief y Clipping de medios. Se presentará la capacidad de Sophia para recopilar, indexar y analizar datos de medios de prensa de manera eficiente, lo que facilita la generación de reportes basados en la percepción mediática. Además, se discutirá cómo los investigadores en ciencias sociales computacionales pueden utilizar Sophia para extraer datasets relevantes y realizar análisis más detallados.

Los reportes en formato PDF, como los Policy Brief y el Clipping de medios, son herramientas valiosas para las instituciones públicas en el análisis de la percepción mediática. Estos reportes proporcionan información resumida y clara sobre una problemática específica y ofrecen recomendaciones para la toma de decisiones. Al utilizar la tecnología Sophia, las instituciones pueden aprovechar su capacidad de recopilación, indexación y análisis de datos de medios de prensa para generar reportes más efectivos. En última instancia, estos reportes ayudan a las instituciones públicas a comprender y abordar de manera más informada los desafíos relacionados con la percepción mediática en el ámbito de las políticas públicas.

3. DIAGNÓSTICO DE LA ARQUITECTURA PREVIA DE SOPHIA

La arquitectura de Sophia ha sido desarrollada de manera continua desde 2020, sin embargo, no se ha seguido una metodología formal de ingeniería de software. El desarrollo de Sophia carece de procesos formales y diagramas. Posteriormente, se han identificado diferentes períodos clave en el ciclo de desarrollo de Sophia, los cuales se resumen en la Figura 4.

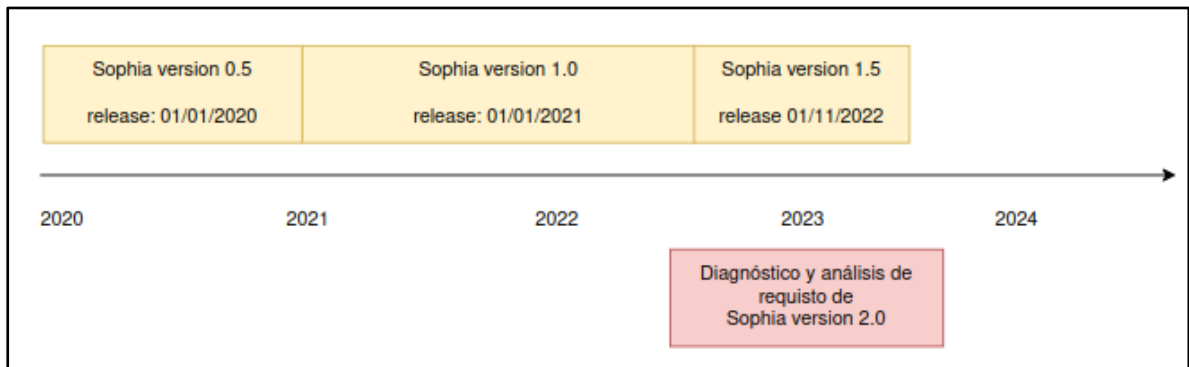


Figura 4: Línea de tiempo de versiones de Sophia.

La primera versión de Sophia (versión 0.5) fue desarrollada específicamente para apoyar la investigación de la tesis de postgrado de Fabian Ruiz. Su objetivo principal era recopilar datos de fuentes de noticias en Chile utilizando una arquitectura simple pero efectiva.

En la segunda versión de Sophia (versión 1.0), se ha mejorado la arquitectura para hacerla escalable y desplegable, lo que significa que ahora es capaz de recopilar datos de medios de prensa en cualquier país. Esta actualización permite a Sophia expandir su alcance y recopilar noticias de diversas fuentes en diferentes regiones del mundo.

En la última versión de Sophia (versión 1.5) se identificaron ciertas deficiencias en la arquitectura existente, lo cual requirió mejoras adicionales. Con el objetivo de abordar estas limitaciones, se llevó a cabo un exhaustivo análisis y diagnóstico de la arquitectura con el fin de ir hacia una nueva versión de Sophia (versión 2.0).

3.1 ¿Quiénes son los usuarios de Sophia versión 1 y cómo utilizan el software?

Durante el análisis de la arquitectura inicial de Sophia, se llevó a cabo la elaboración de un primer diagrama (ver Figura 5) con el objetivo de comprender y entender hacia dónde se quería llegar con la arquitectura, qué mejoras se deseaban realizar y cuál era el flujo que se deseaba implementar en las versiones futuras. Este diagrama representa de manera gráfica los diferentes actores y procesos involucrados en el funcionamiento del sistema.

En el diagrama, se pueden observar tanto la arquitectura inicial como la arquitectura a la que se quiere llegar. En ambas arquitecturas se han identificado como actores a un

Analista, un Admin y un Desarrollador, cada uno con sus respectivas actividades. El Analista tiene como actividad “Extraer un dataset de noticias de prensa a partir de keywords y fechas”, lo cual se define como el proceso de extracción de dataset. Por otro lado, el Admin realiza dos actividades: “Lanzar la recopilación de nuevas noticias desde las páginas web de los medios de prensa” y “Probar scrapers existentes”. Finalmente, el Desarrollador tiene como única actividad “Añadir nuevos códigos de scrapers de medios de prensa en un repositorio”. Tanto el Admin como el Desarrollador se encuentran en el proceso de recopilación.

Las tareas se dividen en dos procesos principales: el proceso de extracción de dataset, en el que participa el Analista, y el proceso de recopilación de datos, en el que participan el Admin y el Desarrollador. Aunque la lógica del caso de uso para esta arquitectura es funcional, hay que considerar que está diseñada para un equipo reducido y con actividades tediosas y complejas para cada actor.

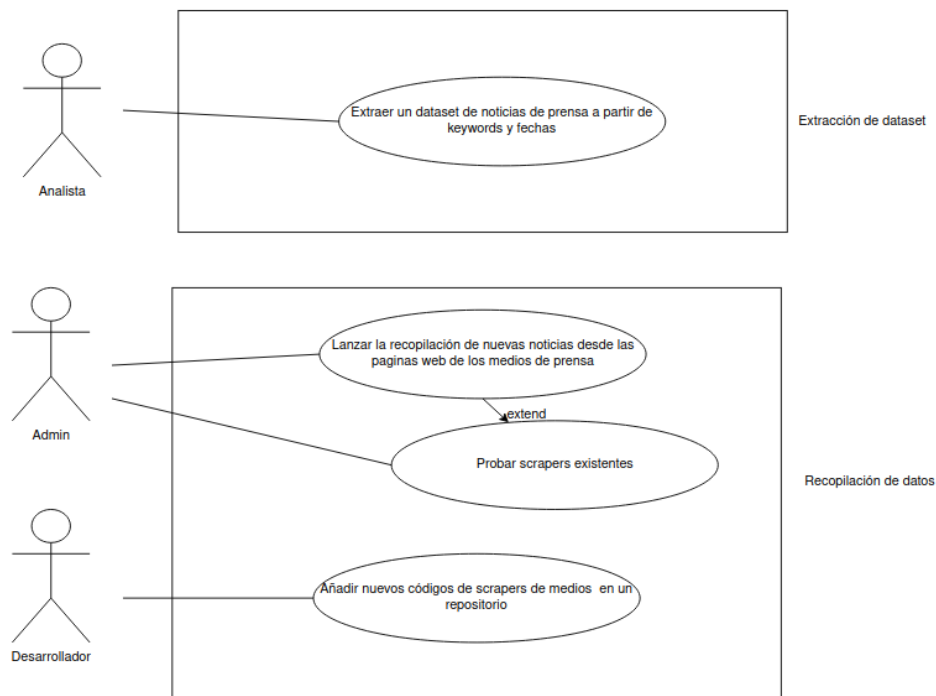


Figura 5: Diagrama de casos de uso de Sophia versión 1.

3.2 ¿Cómo se implementan los casos de uso en Sophia versión 1?

Para entender cómo se implementan los casos de usos se realizan diagramas de flujo, los cuales ayudan a comprender dónde se está ubicado y cómo funciona la arquitectura actualmente para que con esta información se puedan planificar futuros cambios y mejoras para optimizar su funcionamiento.

Además, los diagramas de flujo pueden ser utilizados como una herramienta de comunicación entre el equipo Sophia y futuros desarrolladores de la arquitectura o para el

futuro del proyecto ya que al comprender el diagrama de flujo se puede entender su funcionamiento y cuáles son los componentes principales de esta arquitectura.

3.3 Caso de uso: lanzar la recopilación de nuevas noticias

En la Figura 6 se encuentra la primera parte de lo que es el diagrama de flujo del caso de uso: “Lanzar la recopilación de nuevas noticias” en la arquitectura previa de Sophia la cual tiene como actividad inicial este mismo nombre. En este primer paso, se establece una conexión con el servidor “ground-control”, que se utiliza para desplegar la arquitectura de Sophia y gestionar aspectos de seguridad. Una vez establecida la conexión, se procede a modificar el script Terraform, que se utiliza para automatizar la creación, administración y configuración de infraestructuras.

Dentro del script Terraform, se pueden definir dos características principales: la selección de los países cuyas noticias se desean recopilar y la elección de una copia de seguridad de la base de datos. Estas características permiten personalizar el proceso de recopilación de noticias y asegurar la integridad de los datos.

Finalmente, se ejecuta el script bash “run_sun_deploy.sh”, que despliega el servidor encargado de descargar una versión filtrada de la base de datos de Sophia. Esta versión contiene únicamente los datos correspondientes a los países seleccionados previamente en el script Terraform.

En caso de que el servidor no se despliegue correctamente, se revisan los registros (logs) del sistema para identificar posibles problemas. Si se encuentra algún bug, este se soluciona y se vuelve a ejecutar el script para continuar con el proceso (ver Figura 6).

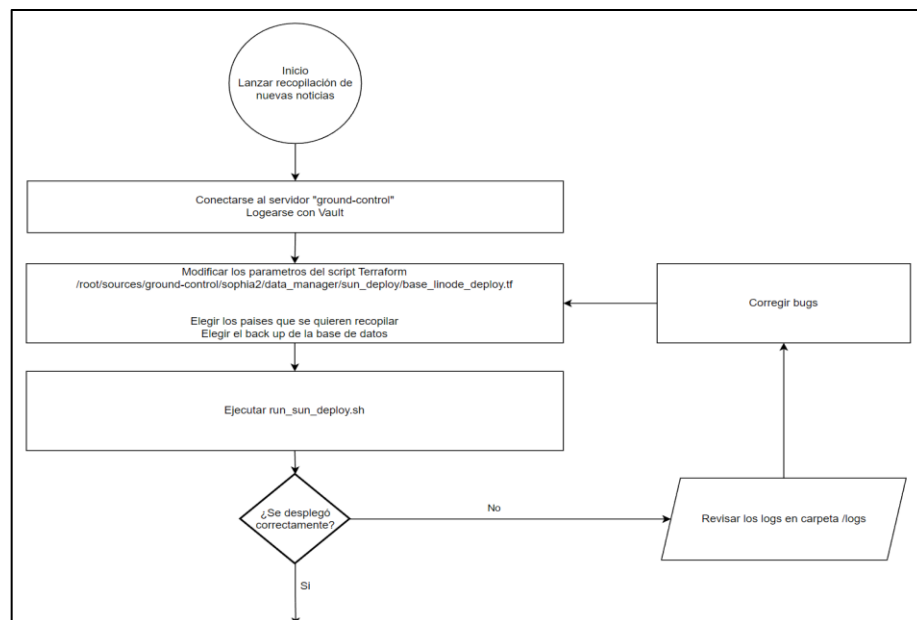


Figura 6: Diagrama de flujo para la recopilación de nuevas noticias en Sophia versión 1. Después de definir los parámetros y obtener una copia de seguridad de la base de datos, se procede a desplegar “Caleuche”, cuya tarea principal es el scraping¹ y crawling² de datos de medios de prensa.

Para llevar a cabo este proceso, se deben realizar modificaciones en los parámetros del script Terraform “base_linode_deploy.tf” y del script “first_start.sh”. En estos scripts, se deben elegir los países, regiones y medios que se desean scrapear. Esto permite personalizar el proceso de scraping y crawling para obtener únicamente los datos relevantes para el proyecto.

Una vez realizadas las modificaciones en los scripts, se ejecuta el script “run_sun_deploy.sh” para desplegar el servidor. Si el servidor no se despliega correctamente, se revisan los registros (logs) del sistema para identificar posibles problemas. Si se encuentra algún bug, este se soluciona y se vuelve a ejecutar el script para continuar con el proceso (ver Figura 7).

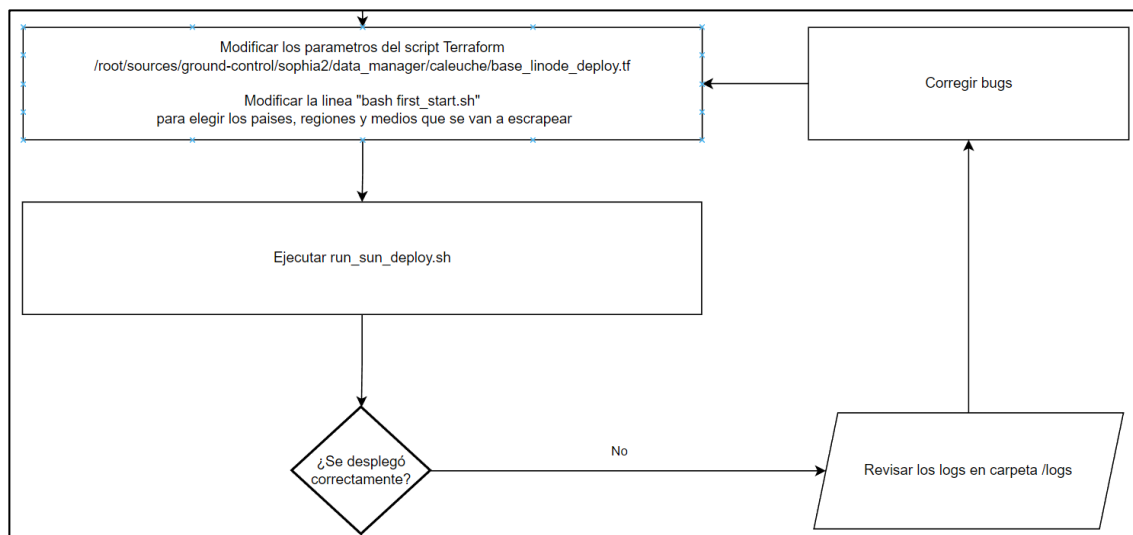


Figura 7: (Continuación)

Para el scraping de noticias que se está realizando en Caleuche, es necesario “Monitorear la recopilación mediante RabbitMQ³” para verificar si la recopilación ha finalizado. En caso afirmativo, se debe apagar manualmente el o los servidores que se utilizaron para la recopilación.

¹ Scraping se refiere al proceso de extracción de datos específicos de sitios web.

² Crawling implica la exploración y navegación automatizada de páginas web para recopilar información.

³ RabbitMQ es un sistema de cola de mensajes que facilita la comunicación asíncrona entre los diferentes componentes de una aplicación.

Después de apagar los servidores, se realiza un monitoreo de la base de datos utilizando Redash⁴. Si se detecta que la recopilación no ha funcionado correctamente, se debe avisar al Scraper Manager para tomar las medidas necesarias. En caso contrario, se puede continuar con el siguiente proceso (ver Figura 8).

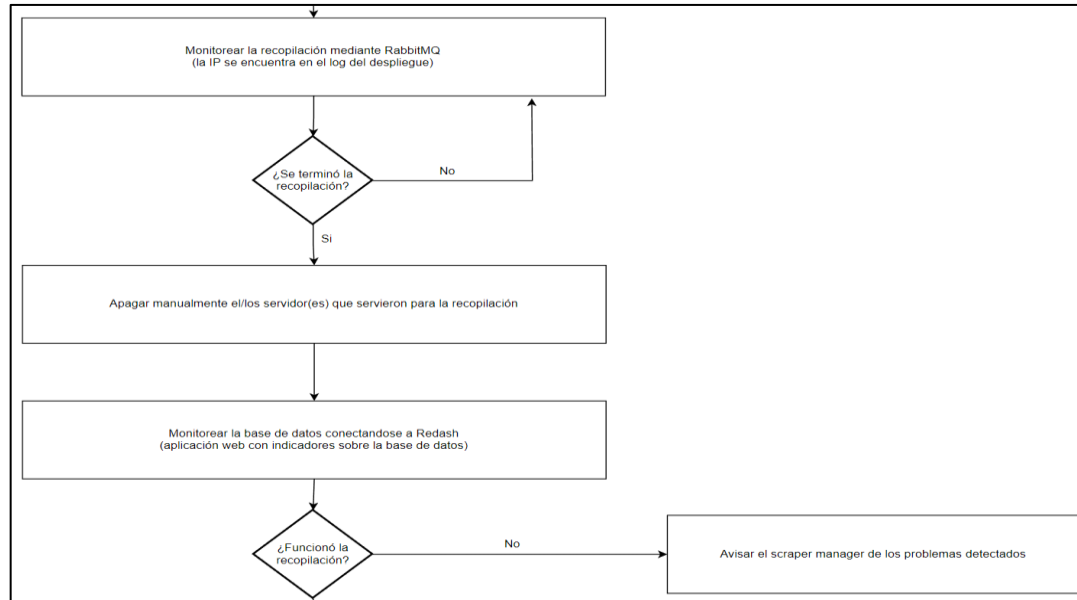


Figura 8: (Continuación)

En caso de que la recopilación de noticias no haya funcionado correctamente, se debe realizar una copia de seguridad de la base de datos utilizando el script “run_caleuche_backup.sh”. Si la copia de seguridad no funciona, se debe revisar el archivo de configuración de Terraform y volver a intentar realizar la copia de seguridad. Si la copia de seguridad se realiza con éxito, se puede continuar con el siguiente proceso (ver Figura 9).

Es importante realizar una copia de seguridad de la base de datos para asegurar la integridad y seguridad de los datos. El script “run_caleuche_backup.sh” automatiza este proceso y facilita la realización del backup. En caso de problemas, se puede revisar el archivo de configuración de Terraform para identificar posibles errores y solucionarlos antes de volver a intentar realizar la copia de seguridad.

⁴ Es una herramienta de visualización y exploración de datos que permite a los usuarios conectarse a diversas fuentes de datos, crear consultas y visualizaciones interactivas, programar actualizaciones y alertas, compartir y colaborar en la interpretación de datos, y garantizar la seguridad de los datos y el control de acceso.

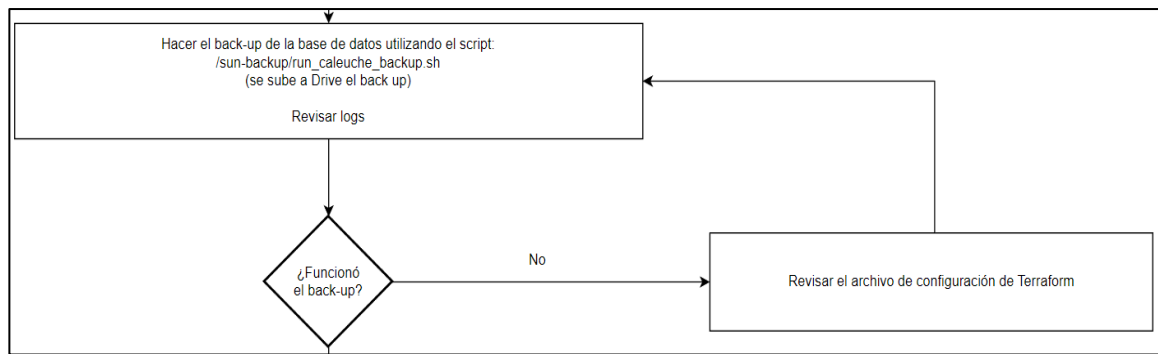


Figura 9: (Continuación)

Después de realizar la copia de seguridad de la base de datos, se debe revisar los parámetros para desplegar el motor de búsqueda. Para ello, se utiliza un nuevo archivo Terraform llamado “base_linode_deploy.tf”. Es importante tener en cuenta que estos archivos de Terraform son utilizados en los distintos servidores que existen, por lo que para cada servidor existe un archivo Terraform con el mismo nombre.

Una vez revisados los parámetros, se procede a levantar el motor de búsqueda de Elasticsearch para indexar los nuevos datos de la base de datos. Para ello, se ejecuta el script “run_sun_search.sh”. Después de indexar los datos, se crea una copia de seguridad de los datos de búsqueda de sun-search utilizando el archivo python “sun_search_backup.py”.

Finalmente, se apaga manualmente el servidor de la base de datos y se llega a la actividad final: “Lanzar recopilación de nuevas noticias”. Este proceso permite actualizar la base de datos con las últimas noticias y preparar el sistema para una nueva recopilación (ver Figura 10).

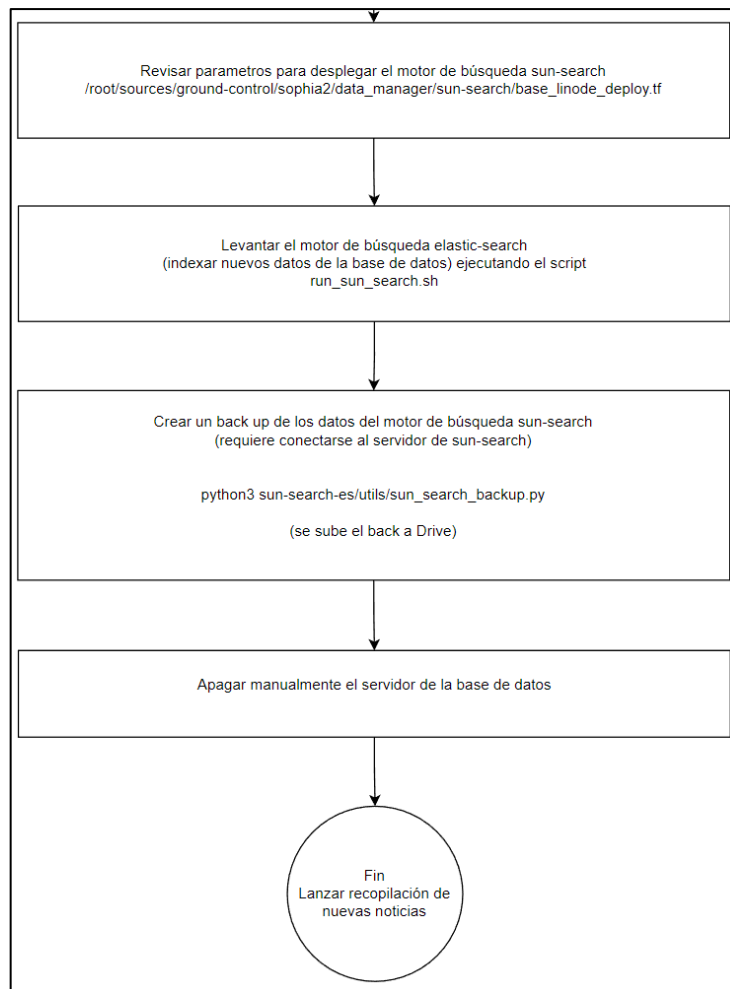


Figura 10: (Continuación)

3.4 Caso de uso: añadir nuevos códigos de scrapers de medios de prensa

En el caso de uso "Añadir nuevos códigos de scrapers de medios de prensa" de Sophia, se muestra como la actividad inicial en la Figura 11. El proceso se inicia conectándose al servidor "ground-control" mediante el uso de Vault. A continuación, se deben realizar modificaciones en los parámetros del script de Terraform, los cuales definen las arañas⁵, para poder ejecutar el scraper. Luego, se procede a ejecutar el script "run_caleuche_test_deploy.sh". En caso de que el script no se haya ejecutado correctamente, se revisan los registros de actividad para identificar si alguna araña ha fallado y determinar las causas correspondientes. Después, se corrigen los errores de las arañas que puedan encontrarse en el script de Terraform, y se realiza una nueva prueba de despliegue. Si el despliegue se ha realizado correctamente, se avanza al siguiente proceso según lo mostrado en la Figura 11.

⁵ Las arañas son scripts o programas que automatizan la tarea de recopilar datos de sitios web de manera eficiente y sistemática.

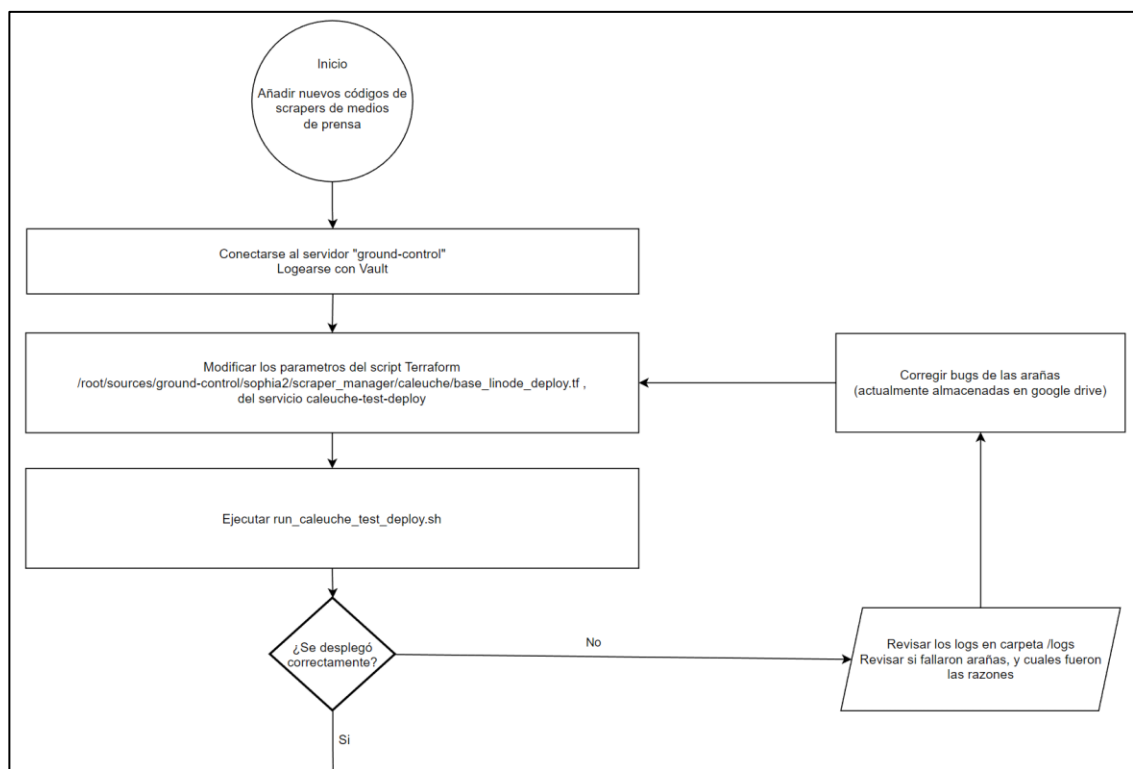


Figura 11: Diagrama de flujo para añadir nuevos códigos de scrapers de medios de prensa en Sophia versión 1.

Una vez que se ha asegurado de que todo esté en condiciones óptimas para la ejecución de las arañas, se procede al siguiente proceso. Este proceso implica la modificación de los parámetros de Terraform y la ejecución del script "run_caleuche_full_deploy.sh". Al igual que en el proceso anterior, si el script no se ejecuta correctamente, se revisan los registros de actividad para corregir los errores correspondientes. En este caso, los errores pueden estar relacionados con la infraestructura. Si el script se ejecuta correctamente, se avanza al siguiente proceso, tal como se muestra en la Figura 12.

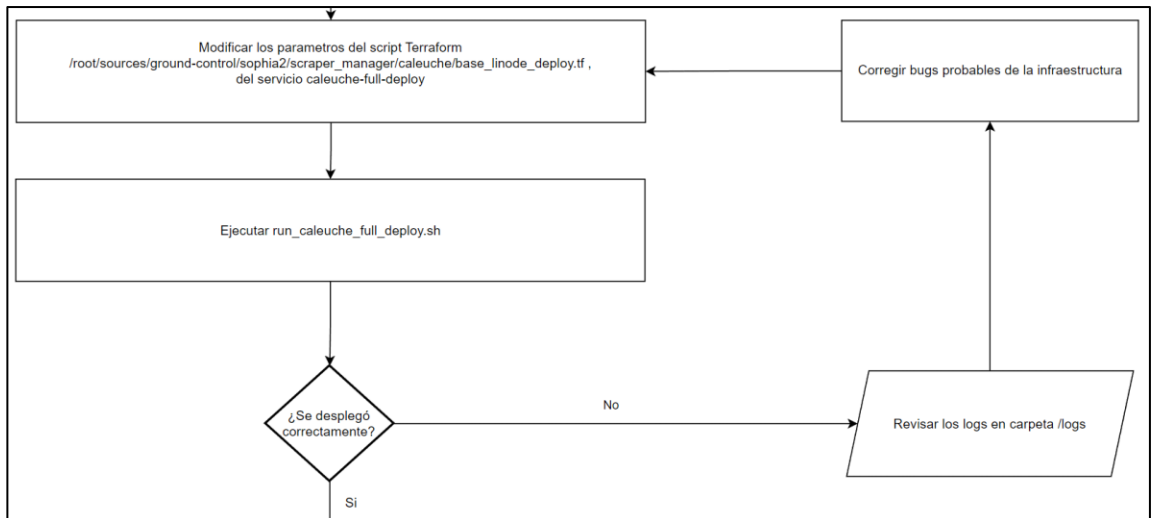


Figura 12: (Continuación)

Para concluir, se notifica al data manager que las nuevas arañas están listas para su despliegue y se le informa sobre cualquier integración fallida. Además, se lleva a cabo una revisión de los servidores para verificar si se apagaron automáticamente. Una vez que se han realizado estas acciones, se considera que la actividad ha finalizado, como se muestra en la Figura 13.

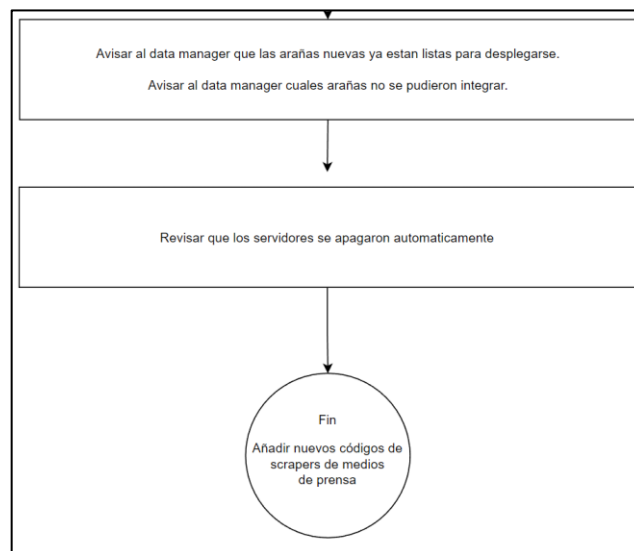


Figura 13: (Continuación)

3.5 Caso de uso: extraer un dataset de noticias de prensa a partir de keywords y fecha

En la versión 1.5, los usuarios tienen la capacidad de extraer conjuntos de datos desde un script de Python disponible en un tutorial de Jupyter Notebook. A continuación, se detalla el proceso:

En primer lugar, se importan las bibliotecas necesarias, como Elasticsearch, y en este caso, las bibliotecas para la visualización de los conjuntos de datos extraídos, utilizando la biblioteca pandas (consultar Figura 14).

```
import elasticsearch
#!pip install elasticsearch==7.16.3

from datetime import datetime
import pandas as pd
from pandasql import sqldf
```

Figura 14: Código Python para la extracción de un dataset.

A continuación, se configuran los parámetros necesarios para utilizar Elasticsearch, incluyendo los medios de prensa que se utilizarán y el país desde el cual se extraerá el conjunto de datos (ver Figura 15)

```
MEDIA_OUTLETS=["rioenlinea","diariopaillaco","diariomafil",
               "diariolaguino","diariolaunion","diariofutrono",
               "diariolagoranco","elnavegante","lapailapaillaco",
               "centralnoticiaspanguipulli","diariodevaldivia",
               "lavozdepaillaco","diariolanco","suractual",
               "noticiaslosrios","diarioribueno","diariosanjose","diariolaguino",
               "diariocorral"]
COUNTRY="chile"
KEYWORD=""
SIMPLE_KEYWORD=True #esta variable es 'True' si KEYWORD es vacio o contiene una palabra simple
```

Figura 15: (Continuación)

En la siguiente ejecución, se realiza la búsqueda de noticias que coincidan con la frase y los parámetros especificados (ver Figura 16)

```

def sophia_search(from_date,to_date,df_list):

    es = elasticsearch.Elasticsearch(
        IP,
        #port=PORT,
        http_auth=(USER, PASS)
    )

    match=""
    if (SIMPLE_KEYWORD):
        match="match"
    else:
        match="match_phrase"

    if len(KEYWORD)==0:
        query = {
            "bool": {
                "filter": [
                    {
                        "range": {
                            "date": {
                                "gte": from_date,
                                "lt": to_date
                            }
                        }
                    },
                    {
                        "term": { "country": COUNTRY }
                    },
                    {
                        "terms": { "media_outlet": MEDIA_OUTLETS }
                    }
                ]
            }
        }
    else:
        query = {
            "bool": {
                "must": [
                    {
                        "match": { "text":KEYWORD}
                    }
                ],
                "filter": [
                    {
                        "range": {
                            "date": {
                                "gte": from_date,
                                "lt": to_date
                            }
                        }
                    },
                    {
                        "term": { "country": COUNTRY }
                    },
                    {
                        "terms": { "media_outlet": MEDIA_OUTLETS }
                    }
                ]
            }
        }

    res = es.search(index="news", query=query, size=10000)
    print("Fechas: " + from_date + "|" + to_date)
    print("Son %d noticias encontradas..." % res['hits']['total']['value'])

```

Figura 16: (Continuación)

Por último, se despliega un dataset generado por pandas para verificar qué noticias fueron encontradas, junto con su respectiva información, como la URL, el título, la fecha, etc. (ver Figura 17)

```

import pandas as pd
data = {'id_news':[], 'country':[], 'media_outlet':[], 'url':[], 'title':[], 'text':[], 'date':[], 'search':[]}

df = pd.DataFrame(data)

for hit in res['hits']['hits']:
    id_news = hit['_source']['id_news']
    country = hit['_source']['country']
    media_outlet = hit['_source']['media_outlet']
    url = hit['_source']['url']
    title = hit['_source']['title']
    text = hit['_source']['text']
    date = hit['_source']['date']
    search = KEYWORD

    new_row = {'id_news':id_news, 'country':country, 'media_outlet':media_outlet, 'url':url, 'title':title, 'text':text, 'date':date, 'search':search}

    df = df.append(new_row, ignore_index=True)
df_list.append(df)

```

Figura 17: (Continuación)

4. DIAGNÓSTICO DE LOS COMPONENTES DE SOPHIA

En esta sección, se presentarán los diferentes diagramas que contribuirán a una comprensión más detallada de la funcionalidad y estructura de la arquitectura de Sophia. Es relevante destacar que el proceso de generación de estos diagramas se llevó a cabo mediante reingeniería inversa, es decir, se realizaron utilizando la arquitectura preexistente, ya que no fueron creados en su momento oportuno. Este enfoque permite una visualización retrospectiva de la arquitectura, identificando sus componentes y relaciones para facilitar la comprensión de su diseño y funcionamiento.

4.1 Ground-control: Centro de pilotaje de Sophia

Ground-control es el componente inicial, este servidor utiliza principalmente las tecnologías Terraform y Vault las cuales juegan un papel fundamental, ya que es el encargado de levantar y establecer toda la arquitectura necesaria. Sin este componente, no sería posible llevar a cabo las demás funcionalidades. Es importante destacar que este servidor no se destruye en ningún momento (ver Figura 18).

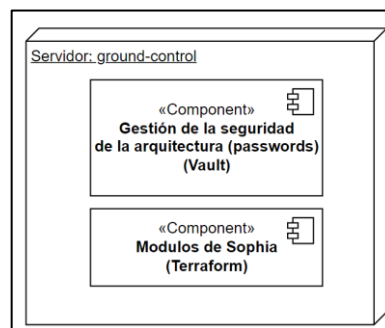


Figura 18: Componente “ground-control” de la versión 1 de Sophia.

4.1.1 ¿Cómo se despliega Sophia versión 1?

En esta sección se muestra en su totalidad como interactúan todos los componentes de la arquitectura de Sophia (ver Figura 19) las cuales en las siguientes secciones se presentarán cada en mayor profundidad.

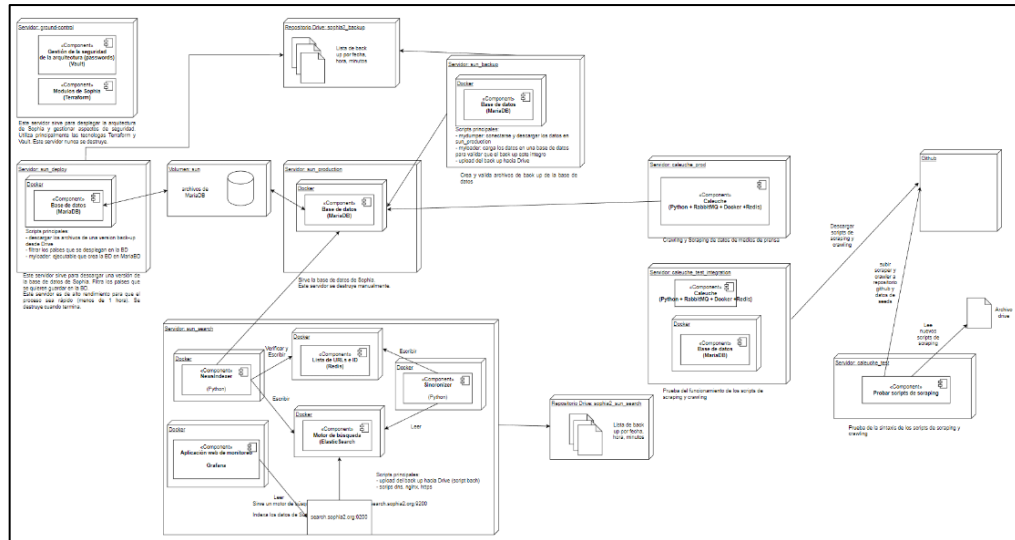


Figura 19: Arquitectura completa de Sophia con sus respectivos componentes.

4.2 Caleuche: Recopilación de noticias de prensa

El servidor Caleuche es el encargado principal de recopilar noticias de prensa mediante técnicas de scraping y crawling de datos (ver Figura 20). Para su funcionamiento, el servidor utiliza diversas tecnologías, como Python, RabbitMQ, Docker y Redis⁶. Además, los datos recopilados se almacenan en una base de datos en MariaDB que se puede ver en la siguiente sección.

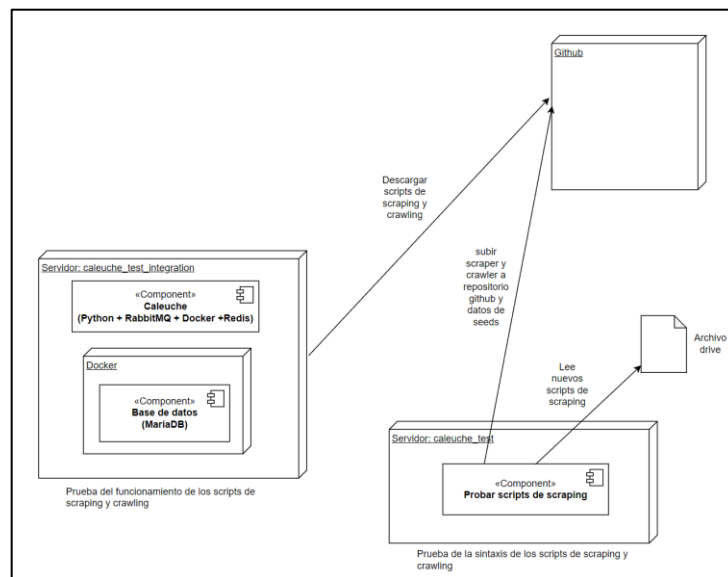


Figura 20: Componente "Caleuche" de la versión 1 de Sophia.

⁶ Es un sistema de almacenamiento en memoria de código abierto diseñado para ofrecer un acceso rápido y eficiente a los datos.

4.2.1 Base de datos Caleuche: Modelo Entidad-Relación

A continuación, se muestra el modelo entidad-relación (ER) de la base de datos Caleuche (ver **¡Error! No se encuentra el origen de la referencia.**) la cual contiene distintos parámetros esenciales que son extraídos y guardados en dicha base. Aquí se pueden encontrar datos necesarios como, por ejemplo:

- media_outlet: medio de prensa de la noticia.
- country: país de la noticia.
- url: enlace de la noticia.
- region: país de la noticia.

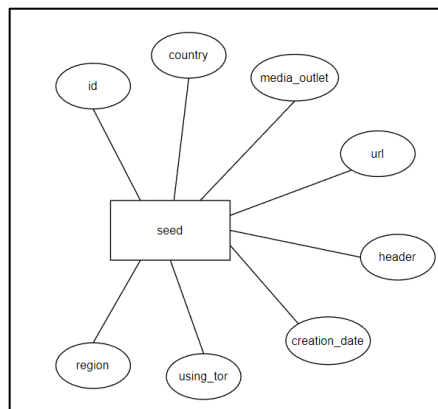


Figura 21: Modelo Entidad-Relación de la base de datos Caleuche

4.2.2 Base de datos Caleuche: Modelo Relacional

A continuación, se presenta el modelo relacional de la base de datos Caleuche el cual contiene los mismos componentes que el modelo Entidad-Relación (ver **¡Error! No se encuentra el origen de la referencia.**).

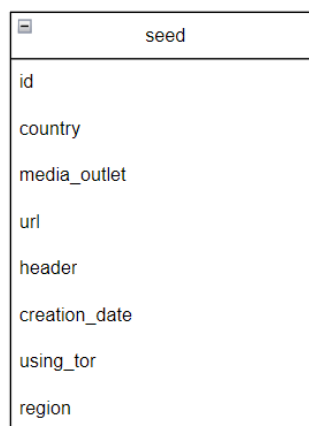


Figura 22: Modelo Relacional de la base de datos Caleuche

4.3 Sun: Almacenamiento de las noticias de prensa

El componente "sun" (ver **¡Error! No se encuentra el origen de la referencia.**) está compuesto por varios servidores, cada uno realizando diferentes acciones que complementan el objetivo principal de almacenar noticias de prensa. A continuación, se describen los servidores que conforman este componente:

1. El servidor "sun_deploy" se encarga de descargar una versión de la base de datos de Sophia y filtrar los países que se desean guardar en la misma. Diseñado para un alto rendimiento, este servidor garantiza que el proceso se realice de manera rápida, en menos de una hora. Una vez finalizado, se destruye automáticamente.
2. El servidor "sun_production" tiene la función de servir la base de datos de Sophia. Este servidor requiere una destrucción manual cuando ya no es necesario.
3. El servidor "sun_backup" se encarga de crear y validar archivos de respaldo de la base de datos. Su función es garantizar la integridad de los datos almacenados.
4. El servidor "caleuche_prod" es el encargado de realizar el scraping y crawling de datos, tal como se explicó anteriormente. Su objetivo principal es recopilar noticias de prensa.
5. Por último, el servidor "sun_search" se encarga de realizar la búsqueda de noticias de prensa. Es responsable de proporcionar un mecanismo de búsqueda eficiente y mantener actualizada la base de datos con las últimas noticias.

Con esta configuración, el componente "sun" cumple un papel integral en la recopilación, almacenamiento y acceso a las noticias de prensa en el sistema Sophia.

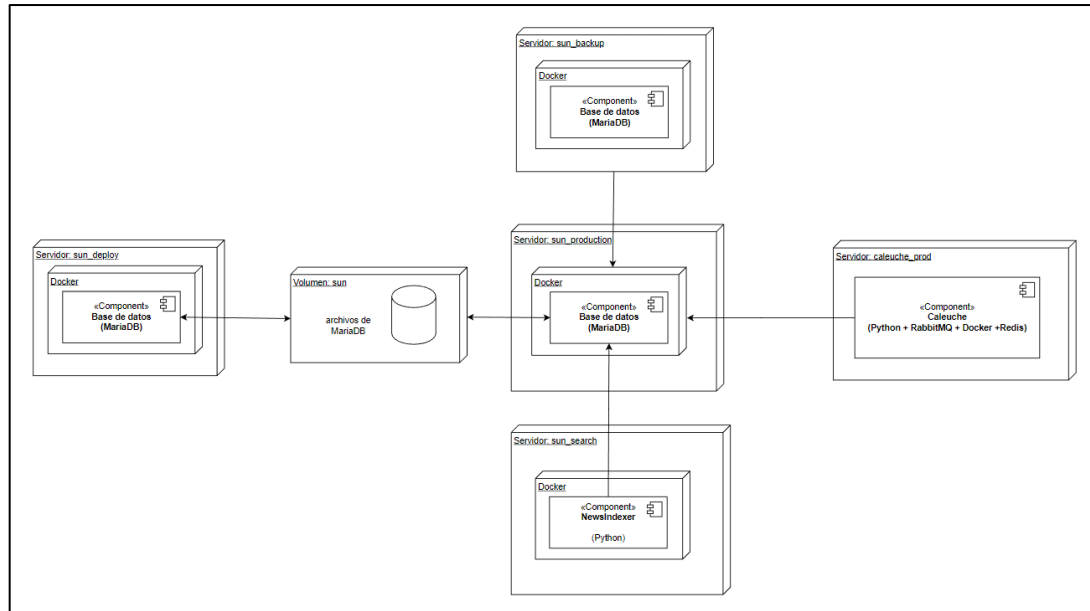


Figura 23: Componente "sun" de la versión 1 de Sophia

4.3.1 Base de datos Sun: Modelo Entidad-Relación

A continuación, se presenta el modelo entidad-relación (ER) de la base de datos Sun la cual contiene la entidad “news” que es a su semejanza a la entidad “seed” que se encuentra en la base de datos Caleuche (ver Figura 24).

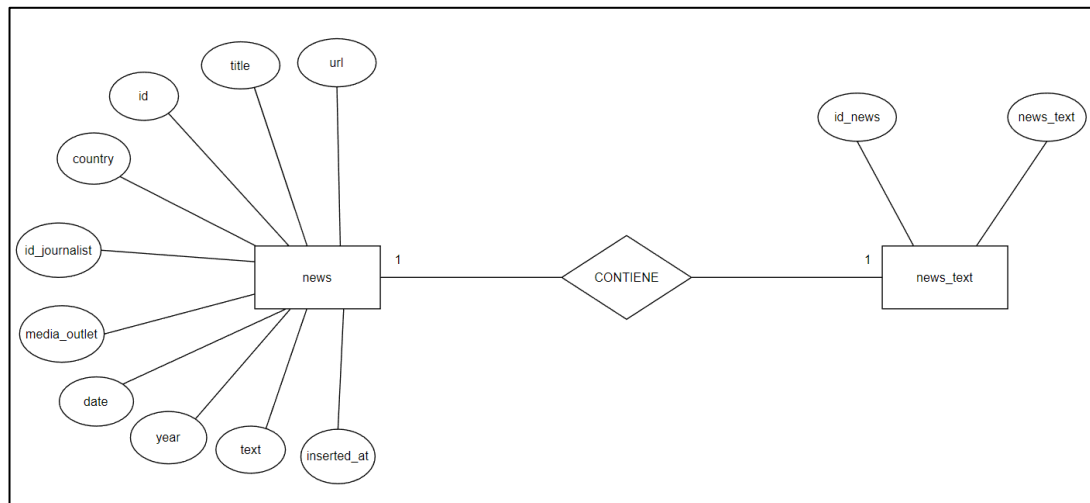


Figura 24: Modelo Entidad-Relación de la base de datos Sun

4.3.2 Base de datos Sun: Modelo Relacional

A continuación, se muestra el modelo relacional la cual es otra manera de representar la base de datos Sun (ver Figura 25).

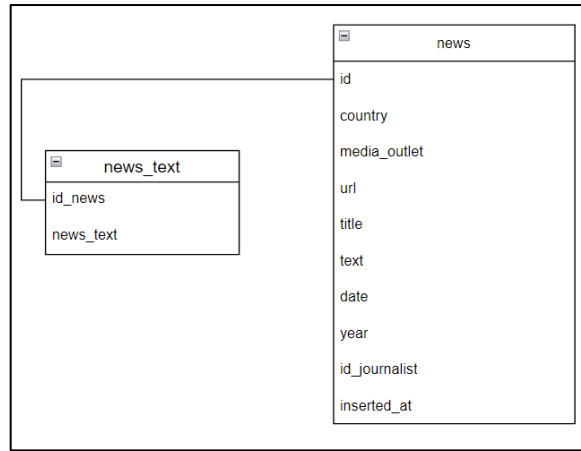


Figura 25: Modelo Relacional de la base de datos Sun

4.4 Sun-search: Búsqueda en las noticias de prensa

En la Figura 26 se muestra el componente "sun-search", el cual se encarga de realizar la búsqueda, recopilación e indexación de nuevas noticias de prensa. Este componente utiliza varias tecnologías, entre las cuales se incluyen Python, Redis y Elasticsearch. Estas tecnologías trabajan en conjunto para realizar las funcionalidades mencionadas.

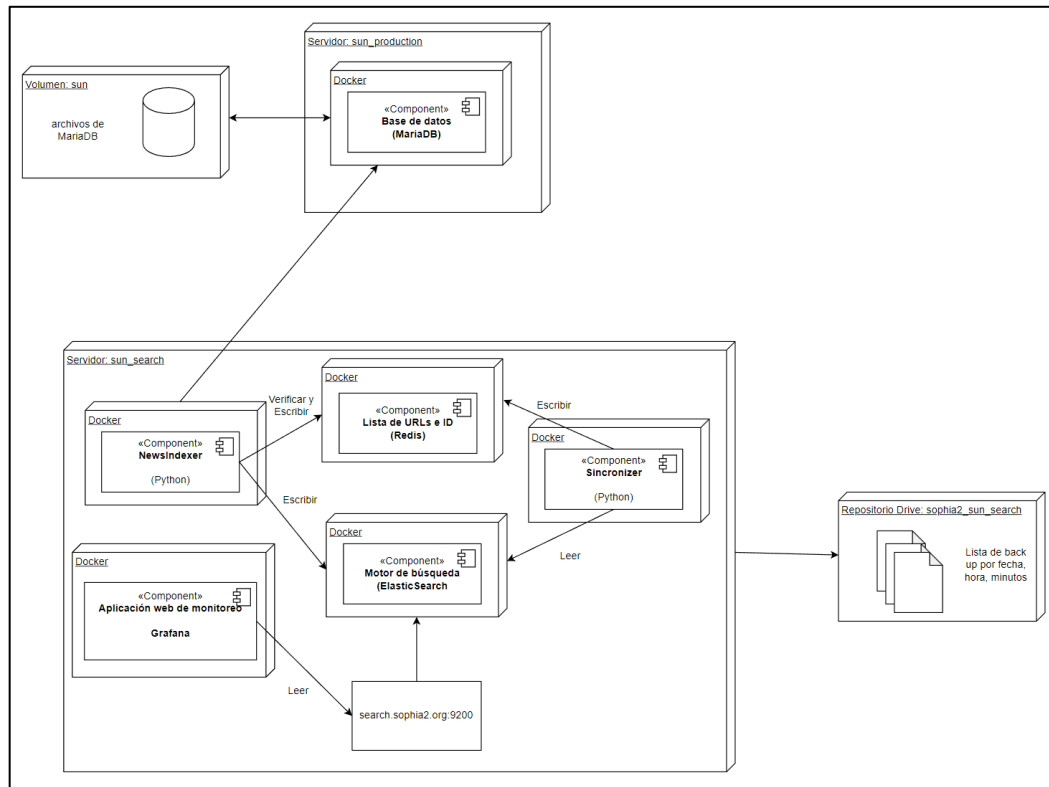


Figura 26: Componente "sun-search" de la versión 1 de Sophia

4.4.1 Motor de búsqueda en documentos: Sun-search

Este sistema utiliza el formato JSON (JavaScript Object Notation) como medio primario para la representación de datos. En el contexto de Elasticsearch, los datos se organizan en lo que se denomina "documentos", cada uno de los cuales es análogo a una entrada individual en una base de datos.

El formato que utiliza el motor de búsqueda en Sun-search es como se muestra en el archivo JSON que se encuentra en el Anexo A.

1. **aliases:** Permite definir alias para el índice, que son nombres alternativos para referirse al mismo índice.
2. **mappings:** Define cómo se deben mapear los campos del índice, especificando los tipos de datos y otras configuraciones.

3. **properties:** Contiene los campos específicos del índice y sus respectivos tipos de datos.
4. **Campos Individuales:** Cada campo dentro de "properties" tiene la siguiente estructura:
 - **type:** Especifica el tipo de datos del campo (text, date, long, etc.).
 - **fields:** Permite definir subcampos o campos adicionales con configuraciones específicas. En este caso, se utiliza para definir un campo adicional de tipo "keyword".
 - **ignore_above:** Indica la longitud máxima de caracteres a indexar para el campo de tipo "keyword". Los valores más largos se truncan.

Cada campo tiene una definición de tipo (text, date, long) y, cuando es aplicable, un campo adicional de tipo "keyword" con la restricción de longitud máxima (ignore_above: 256).

En este caso son los parámetros necesarios de rescatar de las noticias como son country, date, day, id_news, media_outlet, month, text, text_length, title, url y year respectivamente.

5. DISCUSIÓN SOBRE EL DIAGNÓSTICO REALIZADO

En esta sección, se presentarán ejemplos de los análisis previos realizados con el equipo Sophia para comprender mejor la situación actual de la arquitectura. El objetivo es identificar los problemas y/o malas prácticas existentes en el sistema y comenzar a realizar nuevas implementaciones utilizando buenas prácticas arquitectónicas.

5.1 Identificación de problemas (malas prácticas)

Durante el análisis previo con el equipo Sophia, se han identificado una serie de problemas y malas prácticas que afectan la arquitectura actual. A continuación, se describen los problemas específicos relacionados con el diagnóstico realizado.

5.1.1 Falta de proceso formal de diseño arquitectónico

Se ha observado que la arquitectura de Sophia carece de un proceso formal de diseño arquitectónico. Esta falta de definición de usuarios y la ausencia de diagramas arquitectónicos dificultan la comprensión de la arquitectura y la toma de decisiones informadas. Es importante establecer un proceso estructurado que permita definir claramente los componentes, las interacciones y las mejores prácticas arquitectónicas a seguir.

5.1.2 Falta de proceso formal de integración y despliegue continuos

Se ha identificado que no existe un proceso formal de integración y despliegue continuos en Sophia. La falta de un protocolo formal para el ciclo de integración y despliegue continuo puede conducir a la falta de visibilidad, conflictos y retrasos en la entrega del

software. Establecer un proceso formal en esta área facilitará la automatización, mejorará la calidad y agilizará la entrega de nuevas funcionalidades.

5.1.3 Falta de manuales para los usuarios de Sophia

Durante el análisis, se ha detectado que no existen manuales para los usuarios de Sophia. La falta de documentación adecuada en los procesos de indexación, respaldo y restauración de la base de datos puede dificultar la correcta realización de estas tareas, lo que puede llevar a la pérdida de datos y a una recuperación ineficiente en caso de fallos o desastres. Es fundamental desarrollar manuales claros y actualizados que proporcionen orientación detallada para los usuarios de Sophia.

5.1.4 Desorganización en los archivos

Durante el análisis, se han encontrado problemas de desorganización en la estructura de archivos de Sophia. Los problemas identificados incluyen nombres de archivos poco relevantes o explícitos, lo que dificulta la comprensión de su contenido y propósito, así como la presencia de archivos obsoletos que ya no son necesarios en el sistema (ver en la **¡Error! No se encuentra el origen de la referencia.**). Esta desorganización puede afectar la mantenibilidad y escalabilidad de Sophia y requiere una atención inmediata.

```
root@localhost:~/sources/ground-control/sophia2# ls
data_manager      dummy             sun-backup        sun-search-dev
deploy_sun_caleuche.sh  nohup.out         sun-deploy        terraform_playground
discordbot        scraper_manager   sun-search        worldwide
```

Figura 27: Archivos obsoletos en Ground -Control

5.1.5 Problemas en el código (código redundante, credenciales en el código) (Relacionado con la presencia de código redundante y la aparición de credenciales en ciertos archivos)

Durante el análisis, se han identificado problemas en el código de Sophia. Estos problemas incluyen la presencia de código redundante como en la **¡Error! No se encuentra el origen de la referencia.** en donde los conectores de los “secretos” se encuentran reiteradas veces en distintos archivos Python., lo cual dificulta el mantenimiento y aumenta la complejidad del sistema, así como la aparición de credenciales confidenciales en ciertos archivos (ver **¡Error! No se encuentra el origen de la referencia.**), lo cual representa un riesgo de seguridad. Estos problemas deben abordarse para mejorar la calidad, seguridad y legibilidad del código fuente.


```

EARTH_USER_FILE = str(os.environ['EARTH_USER_FILE'])
EARTH_USER_FILE = EARTH_USER_FILE.strip()
EARTH_PASS_FILE = str(os.environ['EARTH_PASS_FILE'])
EARTH_PASS_FILE = EARTH_PASS_FILE.strip()

with open(EARTH_USER_FILE, 'r') as file_credential:
    EARTH_USER = file_credential.read()
with open(EARTH_PASS_FILE, 'r') as file_credential:
    EARTH_PASS = file_credential.read()

EARTH_USER = EARTH_USER.strip()
EARTH_PASS = EARTH_PASS.strip()

SUN_PASS_FILE = str(os.environ['SUN_PASS_FILE'])
SUN_PASS_FILE = SUN_PASS_FILE.strip()

with open(SUN_PASS_FILE, 'r') as file_credential:
    SUN_PASS = file_credential.read()

EARTH_USER_FILE = str(os.environ['EARTH_USER_FILE'])
EARTH_USER_FILE = EARTH_USER_FILE.strip()
EARTH_PASS_FILE = str(os.environ['EARTH_PASS_FILE'])
EARTH_PASS_FILE = EARTH_PASS_FILE.strip()

with open(EARTH_USER_FILE, 'r') as file_credential:
    EARTH_USER = file_credential.read()
with open(EARTH_PASS_FILE, 'r') as file_credential:
    EARTH_PASS = file_credential.read()

EARTH_USER = EARTH_USER.strip()
EARTH_PASS = EARTH_PASS.strip()

SUN_PASS_FILE = str(os.environ['SUN_PASS_FILE'])
SUN_PASS_FILE = SUN_PASS_FILE.strip()

with open(SUN_PASS_FILE, 'r') as file_credential:
    SUN_PASS = file_credential.read()

SUN_PASS_FILE = str(os.environ['SUN_PASS_FILE'])
SUN_PASS_FILE = SUN_PASS_FILE.strip()

with open(SUN_PASS_FILE, 'r') as file_credential:
    SUN_PASS = file_credential.read()

MERCURY_PASS_FILE = str(os.environ['MERCURY_PASS_FILE'])
MERCURY_PASS_FILE = MERCURY_PASS_FILE.strip()
with open(MERCURY_PASS_FILE, 'r') as file_credential:
    MERCURY_PASS = file_credential.read()
MERCURY_PASS = MERCURY_PASS.strip()

```

Figura 28: Código redundante dentro de la arquitectura de Sophia

```

root@localhost:~/sources/ground-control/sophia2/data_manager/caleuche# ls
base_linode_deploy.tf  resources          terraform.tfstate  tfplay
bkp_baselinode         run_caleuche_prod.sh  terraform.tfstate.backup  zipped_resources
logs                   terraform          tfplan

```

Figura 29: Archivos de Componente Caleuche de producción

5.2 Recomendaciones y refactorización prioritaria

A continuación, se presentan una serie de recomendaciones y acciones de refactorización prioritaria para mejorar la eficiencia, la calidad y la mantenibilidad del sistema. Si bien es importante resaltar que estas recomendaciones no serán implementadas en el proyecto actual, es de suma importancia que el equipo Sophia las considere para una mejor implementación de futuros componentes.

1. Definir un protocolo de CI/CD.
2. Desarrollar nuevas funcionalidades utilizando metodología de diagrama 4+1.
3. Refactorizar el componente ground-control.
4. Refactorizar los componentes de Sun y Sun-search.
5. Refactorizar el componente de extracción de datasets para facilitar su uso para usuarios no informáticos.

6. DISEÑO E IMPLEMENTACIÓN DE COMPONENTES DE SOFTWARE PARA LA GENERACIÓN DE REPORTES PDF DE PERCEPCIÓN MEDIÁTICA

Considerando los resultados obtenidos del análisis y la reingeniería aplicada a la arquitectura del software de Sophia, es crucial destacar que el proceso de generación de reportes actual se realiza de forma manual. Tanto la extracción como la generación se hacen de manera directa y poco convencional. Por eso, se proponen mejoras para optimizar y simplificar este proceso.

En este contexto, se presenta una nueva implementación diseñada para facilitar y mejorar significativamente la generación de reportes. Se explorarán diversas opciones durante el desarrollo que serán justificadas.

6.1 Análisis de requisitos y nuevo caso de uso

Uno de los cambios principales e importantes en esta implementación futura es la de las respectivas actividades que realizan los tres actores: Analista, Data Manager y Scraper Manager. El Analista ahora deberá realizar las siguientes cuatro actividades en el orden presentado, el primero será “Extraer un dataset de noticias de prensa a partir de keywords y fechas” con lo cual se podrá “Generar un reporte”, dentro de esta actividad existen dos actividades la cual se debe realizar tanto para “Definir el perímetro de un reporte” para luego “Analizar los datos del reporte”. Todas las actividades que realiza el Analista se encuentran en el proceso de “Extracción de dataset y reportes”.

Luego en el proceso de recopilación de datos existirán dos nuevos actores: Data Manager y Scraper Manager. El Data Manager debe realizar dos actividades, la primera es “Levantar el motor de búsqueda eligiendo los datos indexados” la cual incluye la actividad “Monitorear los datos indexados”, cada vez que el motor de búsqueda sea levantado es necesario “Lanzar la recopilación de nuevas noticias desde páginas web de los medios de prensa” para ir actualizando y obteniendo una mayor cantidad de datos para la generación de reportes.

Por último, se tiene al Scraper Manager el cual debe “Probar scraper existentes” y “Añadir nuevos scrapers” en caso de que algún scraper no este funcionando respectivamente este debe “Notificar los scraper con problemas” la cual está directamente relacionado con la actividad “Monitorear los datos indexados” (ver Figura 30).

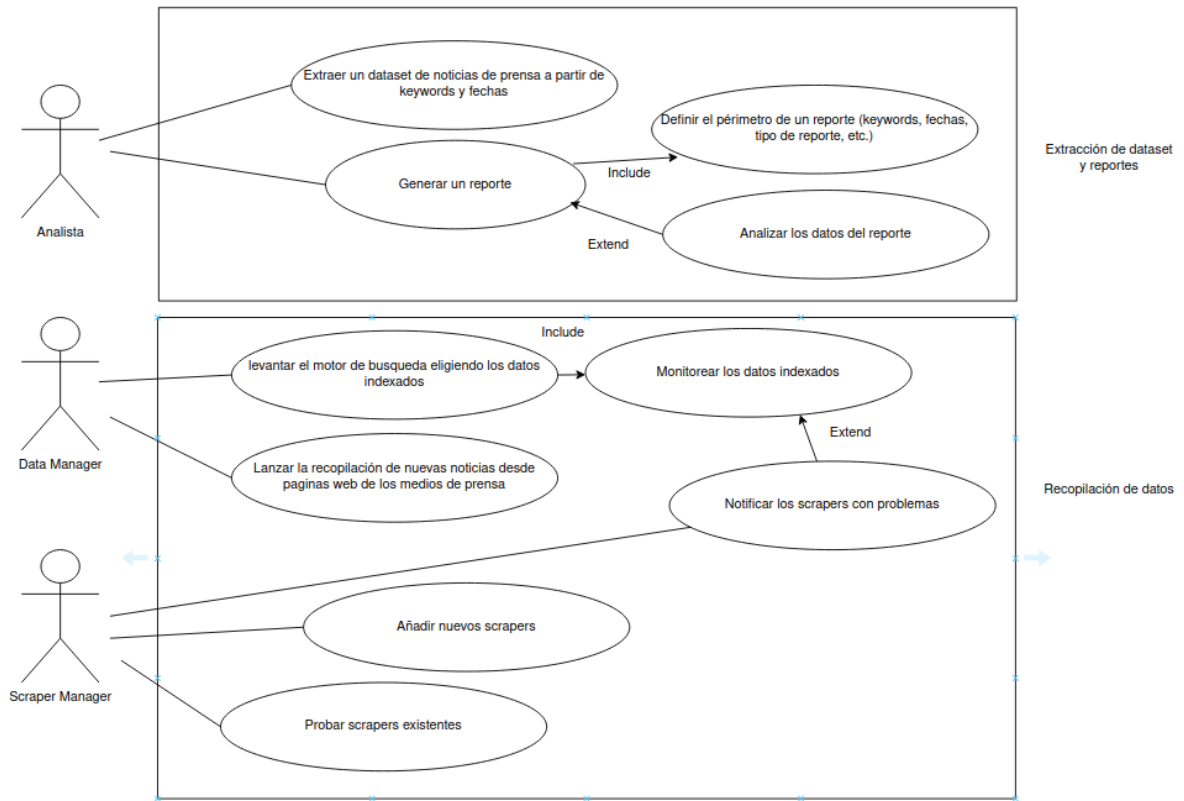


Figura 30: Diagrama de casos de uso de Sophia versión 2.

Como se presentó anteriormente se priorizará en los casos de usos que tienen mayor valor para el futuro de la generación de reportes, es por ello que se dará enfoque a las actividades que están relacionadas al “Analista”.

6.2 Decisiones previas a la implementación de la generación de reportes

Antes de introducir la automatización en la generación de reportes, fue crucial examinar detenidamente las competencias del analista responsable de esta tarea. En un principio, se identificó que este analista contaría con habilidades computacionales básicas, usuales en el día a día. En el transcurso de este proyecto se hizo un análisis más detallado y se proyectó que las habilidades deben evolucionar a nivel competente en el manejo de las herramientas. Es crucial que este analista no solo sea capaz de generar reportes, sino también de contribuir al desarrollo de gráficos futuros.

Con estas premisas en mente, se priorizaron las siguientes consideraciones:

Para el Usuario (Analista):

1. Interfaz Intuitiva y Amigable: Se busca un software con una interfaz intuitiva, asegurando una generación de reportes sin complicaciones técnicas y permitiendo al analista centrarse en su tarea principal.

2. Entrenamiento Sencillo: Se valora la disponibilidad de recursos de capacitación adaptados específicamente para usuarios que estén ingresando a Sophia, facilitando una rápida adopción del software y en un uso efectivo.

Características del Software:

1. Interfaz de Usuario Intuitiva: Se busca un software con una interfaz que simplifique la interacción para usuarios no técnicos, asegurando que las funciones sean fácilmente comprensibles y utilizables.
2. Automatización Sencilla: Se valora la capacidad del software para permitir la automatización de procesos sin requerir conocimientos avanzados de programación, mejorando la eficiencia del analista.
3. Funciones Esenciales y Simples: Se considera la variedad y complejidad de las funciones, dando prioridad a características directas y fácilmente comprensibles para facilitar la generación de reportes sin complicaciones.
4. Actualizaciones Regulares y Desarrollo Activo: Se evalúa la frecuencia de actualizaciones y desarrollo continuo, seleccionando un software con indicadores de longevidad y adaptabilidad a las necesidades cambiantes del analista y la organización.
5. Compatibilidad con Estándares y Facilidad de Integración: Se verifica la conformidad del software con estándares de la industria y su facilidad de integración con otras herramientas y sistemas, garantizando interoperabilidad y la adopción de mejores prácticas.

Dada la urgencia en la necesidad de generar reportes a corto plazo, es esencial buscar un software que permita el desarrollo e implementación de una solución de manera rápida y eficiente. Este software debe cumplir con los criterios previamente establecidos, asegurando una interfaz intuitiva y amigable para el analista, así como funciones esenciales y simples que simplifiquen la generación de reportes. La prioridad radica en encontrar una solución que, a pesar del tiempo limitado, garantice una implementación ágil sin comprometer la calidad y la efectividad en el proceso de generación de reportes.

6.3 Evaluación y elección de Herramientas para Generación Automática

En el proceso de selección para la generación automática de reportes, se evaluaron varias opciones, y la elección final recayó en Jupyter Notebook. La decisión de utilizar esta herramienta se basó en varias razones fundamentales. En primer lugar, su interfaz intuitiva facilita la creación y edición de reportes, lo que lo hace accesible tanto para usuarios técnicos como no técnicos. Además, su capacidad para integrar eficientemente código, texto y visualizaciones proporciona una experiencia de generación de reportes más completa y dinámica.

La elección de Jupyter Notebook también se sustentó en la consideración de las necesidades específicas del analista. La herramienta ofrece una flexibilidad excepcional, permitiendo adaptarse a diversos requerimientos y escenarios de reportes automatizados.

Además, su naturaleza de código abierto respalda el desarrollo continuo y la posibilidad de personalización según las demandas específicas del proceso de generación de reportes.

La comparativa con otras opciones destacó la versatilidad y su robustez, superando las limitaciones de herramientas similares en términos de integración de código, colaboración efectiva y posibilidades de expansión futura. En resumen, la elección para la generación automática de reportes se fundamenta en su capacidad para satisfacer las necesidades actuales del analista y proporcionar un entorno propicio para mejoras y expansiones futuras en este ámbito.

6.4 Implementación de Herramienta para Generación Automática

6.4.1 Implementar Caso de Uso: “Extraer un Dataset”

Como se mencionó previamente, se dio prioridad al segmento del proceso en el que la generación de reportes tendría el mayor impacto, centrándose específicamente en el analista y sus casos de uso (ver Figura 31). En este contexto, el primer caso a abordar implica “extraer un dataset de noticias de prensa a partir de keywords y fechas”. En este sentido, surgen las siguientes interrogantes que se esperan ir desarrollando a lo largo de esta solución:

1. ¿Desde qué fuentes específicas se obtendrá la información?
2. ¿Se ha identificado una lista específica de medios de prensa de los cuales se extraerán las noticias?
3. ¿Cómo se manejarán los cambios en la disponibilidad de información de ciertos medios?
4. ¿Cuál será el intervalo de fechas considerado para la extracción de noticias?
5. ¿Se establecerán fechas específicas de inicio y fin, o se utilizará un rango continuo?
6. ¿Qué regiones geográficas se comprometen en el resultado de la extracción?
7. ¿Se contempla la posibilidad de extracción diferenciada por regiones específicas?
8. ¿Se buscará filtrar la información según categorías específicas, como temas, eventos o tipos de noticias?
9. ¿Existen otros parámetros relevantes que se pretenda obtener durante la extracción de datos?

Como primer paso en el desarrollo de la implementación se debe integrar las distintas librerías que serán de utilidad para la extracción de los datos, esto se realiza al inicio para tener un mayor orden y control de las versiones (ver Figura 31).

Descargar unica vez

```
#!git clone https://github.com/pysentimiento/pysentimiento
#!python3 -m spacy download es_core_news_md
```

Librerías necesarias (Instalar única vez)

```
!pip install tqdm pandas pandasql spacy transformers geopandas matplotlib seaborn xformers openpyxl
!pip install torch torchvision torchaudio
!pip install ipywidgets
```

```
import json
from collections import Counter

#Pandas
import pandas as pd
from pandasql import sqldf

#Spacy
import spacy
from spacy.matcher import PhraseMatcher
from spacy.matcher import Matcher
nlp = spacy.load("es_core_news_md")

#Pandas, Numpy, Matplotlib, Seaborn
import geopandas as gpd
import shapely as shp
import pandas
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import matplotlib.ticker as ticker
from matplotlib.colors import ListedColormap
import matplotlib.colors as colors
```

Figura 31: Código de importación de librerías y paquetes necesarios.

Luego se ingresarán los parámetros necesarios para obtener las noticias que se analizarán. El primer paso es la autenticación en Elasticsearch (ver Figura 32), utilizando las credenciales correspondientes al entorno de Sophia. Este proceso de inicio de sesión es crucial para garantizar el acceso autorizado y seguro a la plataforma.

A continuación, se requiere la definición de los parámetros cruciales para la extracción de noticias. Estos son los siguientes:

1. **COUNTRY:** Se debe indicar el país del cual se extraerán las noticias para la extracción de noticias.
2. **TOPIC:** Se debe especificar el tema que se desea analizar durante la extracción de noticias.
3. **KEYWORDS:** Se debe proporcionar las palabras clave que se buscarán tanto en el título como en el texto de la noticia.

```

HOSTS = "http://search.sophia2.org:9200"
USER= "mvernier"
PASS= "Bfg737LCff"

COUNTRY = "chile"
TOPIC = "Turismo"
KEYWORDS = ["feria", "visitantes", "alojamiento", "actividades turísticas",
"transporte", "gastronomía", "ecoturismo", "cultura local", "guía turístico",
"parques", "patrimonio", "agencia de viajes", "tour", "turista", "turistas",
"aventura", "termas", "gastronomía", "reserva",
"senderismo", "restaurant", "trekking", "parque", "esquí", "atractivos", "eventos",
"hospedaje", "panoramas", "restaurante", "viaje",
"hotel", "playa", "vacaciones", "alojamiento", "turismo", "artesanía", "turísticos", "cultura"]

```

Figura 32: Código de los parámetros para la búsqueda en Sophia.

En la siguiente ejecución, se busca obtener la región en la cual se obtendrán las comunas correspondientes. Con el propósito de simplificar el proceso para el analista, se ha empleado la librería ipywidgets. Esto permite la creación de una lista desplegable que se presenta en la Figura 33, facilitando la selección de la región de interés.

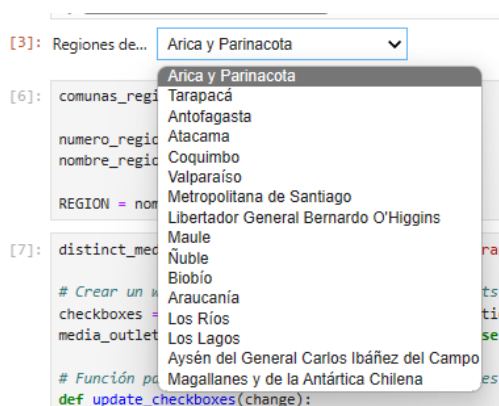


Figura 33: Widget de selección de región.

A continuación, es necesario seleccionar los medios de prensa de los cuales se extraerán las noticias. Al igual que en la situación anterior, se ha incorporado el uso de la misma librería, proporcionando un checklist que presenta todos los medios de prensa disponibles para su selección, como se muestra en la Figura 34.

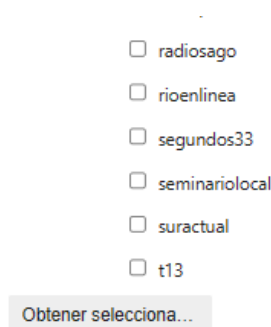


Figura 34: Widget de selección de medios de prensa.

Luego se procederá a seleccionar el rango de fechas para la extracción de noticias. Para mejorar la experiencia del usuario, especialmente la del analista, se ha recurrido nuevamente a la utilización de la misma librería. Esta herramienta facilita la selección de fechas mediante un selector específico, optimizando así la interacción y personalización del rango temporal deseado (ver Figura 35).

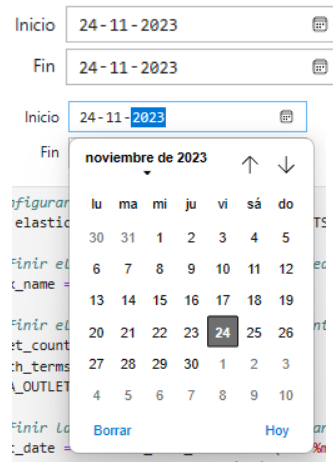


Figura 35: Widget de selección de fechas (inicio y termino).

Hasta este punto, se seleccionaron los parámetros necesarios para extraer noticias. Ahora, se busca noticias en la base de datos que las contiene. En este apartado, el analista simplemente debe ejecutar la operación, y la búsqueda se realizará automáticamente según los parámetros proporcionados. Se llevará a cabo un análisis para determinar si las noticias cumplen con los criterios de estar en el país especificado (COUNTRY), si provienen de los medios de prensa seleccionados (MEDIA_OUTLETS), y si se encuentran dentro del rango de fechas establecido (ver Figura 36)


```

# Configurar la conexión a Elasticsearch
es = elasticsearch.Elasticsearch(hosts=HOSTS, basic_auth=(USER, PASS))

# Definir el nombre del índice del que deseas obtener el recuento
index_name = "news"

# Definir el país específico que deseas contar
target_country = COUNTRY
search_terms = KEYWORDS
MEDIA_OUTLETS = selected

# Definir las fechas de inicio y fin del rango
start_date = selected_date_1.strftime("%Y-%m-%d")
end_date = selected_date_2.strftime("%Y-%m-%d")

# Crear una consulta que combina el término de búsqueda para el país y el rango de fechas
query = {
    "bool": {
        "must": [
            {
                "term": { "country": COUNTRY },
                "terms": { "media_outlet": MEDIA_OUTLETS },
            },
            {
                "range": {
                    "date": {
                        "gte": start_date,
                        "lte": end_date
                    }
                }
            }
        ]
    }
}

# Realizar la consulta de conteo
count_result = es.count(index=index_name, query=query)

# Obtener el recuento de documentos relacionados con el país y dentro del rango de fechas
document_count = count_result["count"]

print(f'Hay {document_count} noticias encontradas en {target_country.capitalize()} en el rango de fechas {start_date} a {end_date}.')

```

Figura 36: Código de búsqueda de noticias dentro de Elasticsearch.

Después de rescatar las noticias que cumplen con los parámetros establecidos, se almacenan en un dataset que se utilizará posteriormente para llevar a cabo diversos tipos de análisis (ver Figura 37).

```

import pandas as pd
data = {'id_news':[], 'country':[], 'media_outlet':[], 'url':[], 'title':[], 'text':[], 'date':[]}

# Crear un DataFrame vacío
df = pd.DataFrame(data)

# Contador de noticias obtenidas
news_count = 0

while len(hits) > 0:
    for hit in hits:
        id_news = hit['_source']['id_news']
        country = hit['_source']['country']
        media_outlet = hit['_source']['media_outlet']
        url = hit['_source']['url']
        title = hit['_source']['title']
        text = hit['_source']['text']
        date = hit['_source']['date']

        new_row = {
            'id_news': id_news,
            'country': country,
            'media_outlet': media_outlet,
            'url': url,
            'title': title,
            'text': text,
            'date': date
        }

        df = pd.concat([df, pd.DataFrame([new_row])], ignore_index=True)

        news_count += 1
        print(f"Noticias obtenidas: {news_count}/{document_count}", end="\r") # Imprimir progreso en la misma línea

# Realizar una solicitud de Scroll para obtener el siguiente lote
res = es.scroll(scroll_id=scroll_id, scroll=scroll_time)
hits = res['hits']['hits']

# Crear un DataFrame a partir de los resultados
df = pd.DataFrame(df)

print("\nTotal de noticias encontradas:", df.shape[0])

```

Figura 37: Código de guardado de noticias encontradas en un Dataset.

Una de las principales complicaciones encontradas durante la extracción de datos era la duplicidad de noticias. Esto se debe a que algunos medios de prensa publican la misma noticia. Para evitar interferencias en los resultados esperados, se implementa la eliminación de todas las noticias duplicadas en el dataset, considerando duplicados aquellos que compartan el título, texto o URL (ver figura 38) luego se guarda en formato CSV para su uso posterior.

```
df = df.dropna(subset=['text'])
df = df.dropna(subset=['title'])
df = df.drop_duplicates(subset='url', keep='first')
df = df.drop_duplicates(subset='title', keep='first')
df = df.drop_duplicates(subset='text', keep='first')
df = df.reset_index(drop=True)

# Nombre del archivo CSV que se guardará en el directorio actual
nombre_archivo_csv = f'noticias_{nombre_region}_{start_date}_{end_date}.csv'

# Guardar el DataFrame en un archivo CSV en el directorio actual
df.to_csv(nombre_archivo_csv, index=False)

df
```

Figura 38: Código de limpieza del dataset.

Tras la depuración del conjunto de datos, se incorporan las coincidencias de palabras clave identificadas dentro de las noticias recuperadas. A continuación, se lleva a cabo una clasificación descendente de estas coincidencias según su frecuencia. Esta etapa facilita la presentación y el análisis, permitiendo evaluar la adecuación del modelo y la concordancia con las expectativas predefinidas (ver Figura 39).

```
# Función para calcular las palabras clave y el score de relevancia
def calculate_keywords_and_score(text):
    words = text.lower().split()
    found_keywords = [word for word in words if word in KEYWORDS]
    score = len(found_keywords)
    return found_keywords, score

# Aplicar la función a la columna 'text'
df[['found_keywords', 'relevance_score']] = df['text'].apply(calculate_keywords_and_score).apply(pd.Series)

df = df.sort_values(by='relevance_score', ascending=False)
df = df.reset_index(drop=True)

df

#
```

Figura 39: Código para asignar y ordenar el dataset dada las palabras claves.

Después, se agregan columnas de ceros, tantas como comunas, para buscar por cada noticia y determinar si alguna comuna se menciona. Si se menciona una comuna, se le asignará el valor uno; si no, se mantendrá con el valor cero (ver Figura 40).

```

CITIES = comunas_region

matcher_cities = PhraseMatcher(nlp.vocab)

for city in CITIES:
    matcher_cities.add(city, [nlp(city)])
    df.insert(7,city.replace(" ","_"),0)

df.columns.values

for index,row in df.iterrows():
    print(index)
    txt = row["text"]

    try:
        doc = nlp(txt)
        matches_cities = matcher_cities(doc)

        for match_id, start, end in matches_cities:
            span = doc[start:end] # The matched span
            df.at[index,span.text.replace(" ","_")]=1

    except Exception as e:
        print(e)
        pass

```

Figura 40: Código para etiquetar las comunas mencionadas.

Posteriormente, mediante la librería Pysentimiento, se añade una nueva columna para realizar un análisis de las noticias rescatadas, determinando su tonalidad positiva o negativa. Se asigna un valor decimal entre 0.0 y 1.0, donde una cercanía al 1.0 indica una tonalidad positiva, y a medida que se acerca al 0.0, la noticia es percibida como negativa, luego se guarda en formato CSV para su uso posterior (ver Figura 41).

```

#Agregar columnas a los datasets
df['prediction'] = ""
df['score'] = ""

for index, row in tqdm(df.iterrows(), desc='Análisis de sentimiento', total=df.shape[0]):

    sentiment_value = sentiment_pipeline(row['title'])

    # Insertamos en dataframe
    df.at[index, "prediction"] = sentiment_value[0].get('label')
    df.at[index, "score"] = sentiment_value[0].get('score')

df

# Nombre del archivo CSV que se guardará en el directorio actual
nombre_archivo_csv = f"noticias_{REGION}_por_comuna_{start_date}_{end_date}.csv"

# Guardar el DataFrame en un archivo CSV en el directorio actual
df.to_csv(nombre_archivo_csv, index=False)

```

Figura 41: Código para asignar porcentaje de sentimiento a las noticias.

Finalmente, todas las variables utilizadas durante el proceso de extracción de noticias se guardan para su posterior utilización en el siguiente caso de uso del analista. Esta

información se almacena en un formato JSON y se termina el contenido de este primer Notebook (ver Figura 42).

```
data = {
    "COUNTRY": COUNTRY,
    "REGION": REGION,
    "NUMERO_REGION": numero_region,
    "COMUNAS": comunas_region,
    "MEDIA_OUTLETS" : selected,
    "TOPIC" : TOPIC,
    "KEYWORDS": KEYWORDS,
    "START_DATE": start_date,
    "END_DATE": end_date
}

with open(f"datos.json", "w") as json_file:
    json.dump(data, json_file)
```

Figura 42: Código para guardar variables en un archivo JSON.

6.4.2 Implementar Caso de Uso: “Generar un reporte”

Prosiguiendo con las tareas asignadas al analista, es crucial comprender la implementación del "Generar un reporte", cuyo objetivo principal consiste en analizar los datos incluidos en el reporte. Estos datos se refieren a los gráficos y tablas que se pueden extraer durante el análisis. Aunque se espera abarcar la mayoría de los gráficos que normalmente contendría un reporte, es fundamental considerar que este proceso de análisis puede extenderse y no abarcar todas las variables.

Al igual que el primer Notebook se presentan los archivos y librerías necesarias al inicio para instalarlas y mantener un mayor control de las versiones (ver Figura 43).

Descargar unica vez

```
#!git clone https://github.com/pysentimiento/pysentimiento
#!python3 -m spacy download es_core_news_md
```

Librerías necesarias (Instalar única vez)

```
!pip install tqdm pandas pandasql spacy transformers geopandas matplotlib seaborn xformers openpyxl
!pip install torch torchvision torchaudio
!pip install ipywidgets
```

```
import json
from collections import Counter

#Pandas
import pandas as pd
from pandasql import sqldf

#Spacy
import spacy
from spacy.matcher import PhraseMatcher
from spacy.matcher import Matcher
nlp = spacy.load("es_core_news_md")

#Pandas, Numpy, Matplotlib, Seaborn
import geopandas as gpd
import shapely as shp
import pandas
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import matplotlib.ticker as ticker
from matplotlib.colors import ListedColormap
import matplotlib.colors as colors
```

Figura 43: Código de importación de librerías y paquetes necesarios.

Luego se deben obtener los datos utilizados como parámetros en el anterior notebook, por eso se carga con sus respectivos nombres de variables (ver figura 44) esto con el fin de poder generar de manera más dinámica los gráficos que serán generados en este notebook.

```
with open("datos.json", "r") as json_file:
    loaded_data = json.load(json_file)

COUNTRY = loaded_data["COUNTRY"]
REGION = loaded_data["REGION"]
NUMERO_REGION = loaded_data["NUMERO_REGION"]
COMUNAS = loaded_data["COMUNAS"]
MEDIA_OUTLETS = loaded_data["MEDIA_OUTLETS"]
KEYWORDS = loaded_data["KEYWORDS"]
START_DATE = loaded_data["START_DATE"]
END_DATE = loaded_data["END_DATE"]
```

Figura 44: Código de importación de librerías y paquetes necesarios.

Según avanza este notebook, se presentan varios tipos de gráficos que usan diferentes parámetros y datos obtenidos en el notebook anterior. En consecuencia, a continuación, se exhibirán los gráficos más esenciales y útiles que surgieron durante el desarrollo.

Uno de los métodos mas esenciales fue la utilización de mapas de calor geográficamente en donde las comunas fuesen un valor que se representa visualmente con respectivos colores. En este caso se corrobora que tipo de región se está extrayendo los datos para luego proyectar la figura necesaria dado los datos ya extraídos. En el siguiente código se muestra el método para el mapa de calor geográfico (ver figura 45).

```
SHAPEFILE=r"./shape/comunas.shp" #shapefile de las regiones y comunas de Chile
# fuente: https://www.bcn.cl/sit/mapas_vectoriales

territory = gpd.read_file(SHAPEFILE, encoding="utf-8")
territory = territory[territory["codregion"] == NUMERO_REGION]
# Obtenemos los datos de las Columnas "Comuna", "geometry"
territory = territory[["Comuna", "geometry"]]

from matplotlib.patheffects import withStroke, Normal

def save_map(df, territory, image_name: str = "Mapa_Calor"):
    # Junta las tablas df y territory
    territory = territory.merge(df, on="Comuna")

    # Inicializa matplotlib para la creación del mapa
    fig, ax = plt.subplots(figsize=(10, 10))
    max_presente = df["Valor"].max() # Valor máximo para la barra de color
    territory.plot(column="Valor", ax=ax, edgecolor="gray", cmap="OrRd", legend=False).set_axis_off()

    # Agrega etiquetas de comunas en el mapa con sombra
    for i, row in territory.iterrows():
        comuna = row["Comuna"]
        porcentaje = row["Valor"]
        x = row["geometry"].centroid.x
        y = row["geometry"].centroid.y

        # Determinar el color de la etiqueta según el color del fondo
        fondo = row["Valor"]

        # Usar un color de etiqueta más brillante en áreas de fondo oscuro
        etiqueta_color = 'white' # Fondo oscuro, etiqueta clara

        # Agregar comuna y porcentaje en la etiqueta con sombra
        etiqueta_text = f'({comuna})\n(porcentaje:2%)' # Formatear el porcentaje como texto
        sombra = withStroke(linewidth=3, foreground="gray") # Configurar la sombra
        ax.text(x, y, etiqueta_text, fontsize=10, ha='center', va='center', color=etiqueta_color, path_effects=[sombra, Normal()])

    bar_info = plt.cm.ScalarMappable(cmap="Reds", norm=plt.Normalize(vmin=0, vmax=max_presente))
    bar_info._A = []
    cbar = fig.colorbar(bar_info)
```

Figura 45: Código para generación de mapas de calor geográfico.

Con este método se pueden obtener distintos mapas de calor, a continuación, se presenta un ejemplo de ello en la Figura 46.

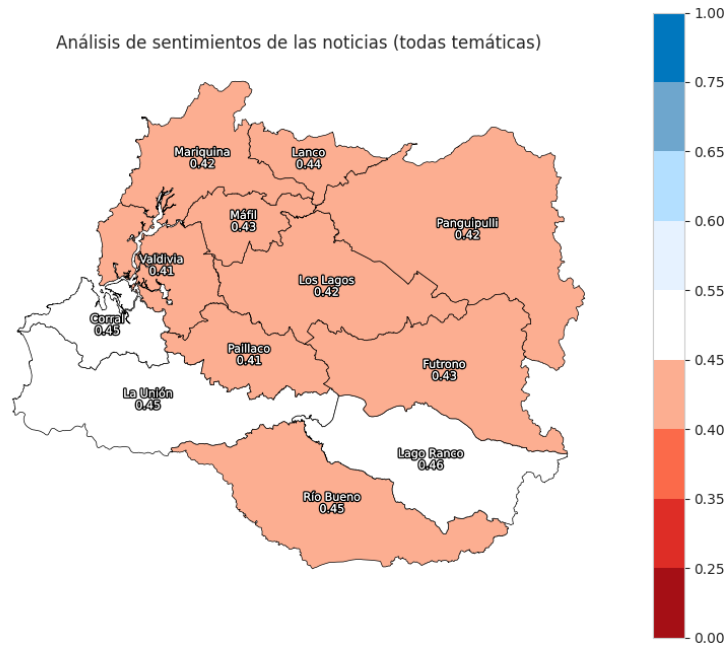


Figura 46: Mapa de calor geográfico respecto al análisis de sentimientos.

Dentro de este notebook se pueden encontrar una variedad de gráficos que se pueden extraer y generar para los reportes, alguno de ellos son los que se ven a continuación (ver Figura 47 y Figura 48).

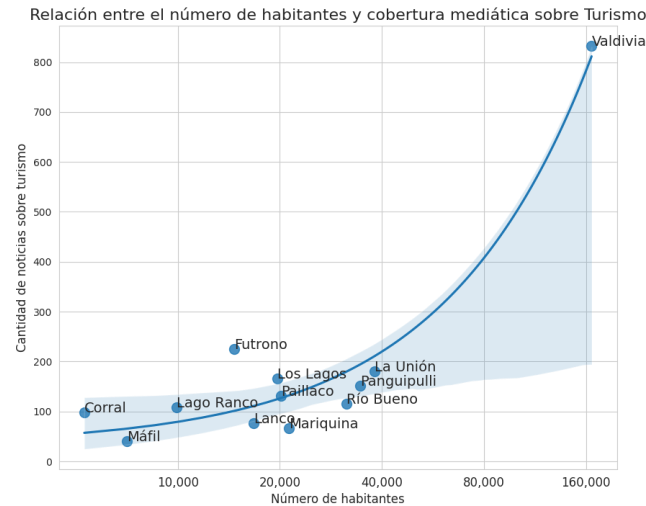


Figura 47: Gráfico de relación entre número de habitante y cobertura mediática sobre Turismo.

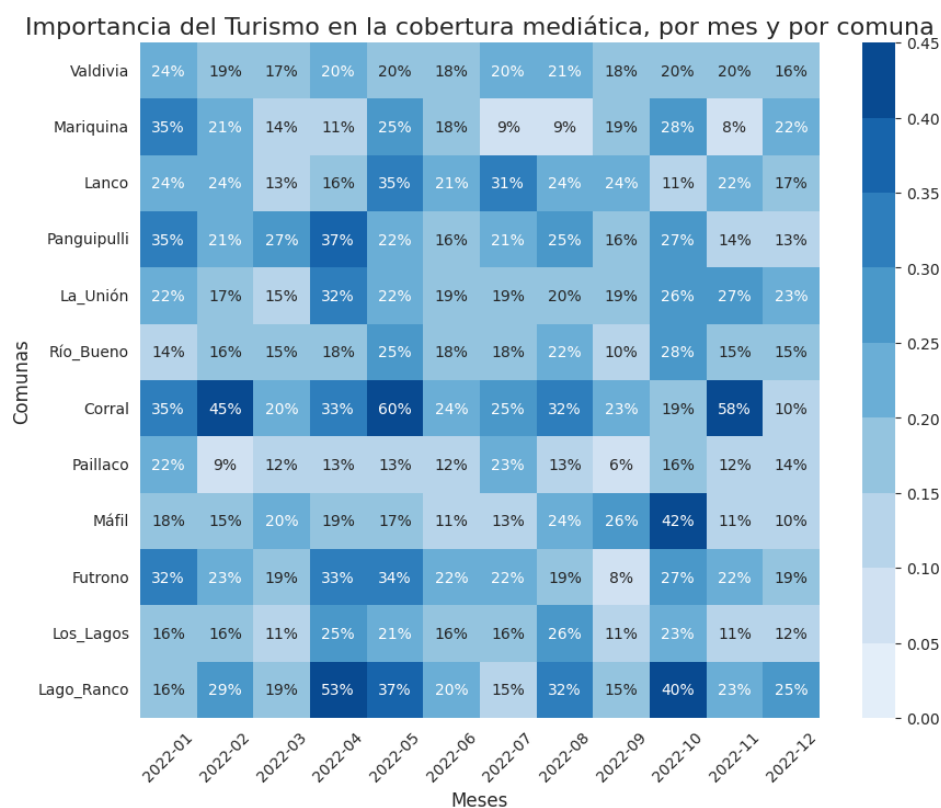


Figura 48: Mapa de calor respecto a la importancia sobre la temática, comunas y mes.

6.5 Implementación de los resultados del Análisis.

Finalmente, el analista deberá asignar y generar el reporte, utilizando los gráficos necesarios según las especificaciones del cliente y los objetivos buscados en el reporte solicitado. Para ello, cada gráfico se guardará localmente en el dispositivo.

7. DISCUSIÓN Y CONCLUSIÓN

En esta sección, se realiza una inmersión en una detallada discusión y conclusión, explorando minuciosamente cada etapa del proceso llevado a cabo en el proyecto. La atención se centra en las decisiones estratégicas adoptadas, profundizando en la comprensión del proceso y evaluando su efectividad. Este análisis se ejecuta considerando las necesidades específicas de la empresa, brindando una visión integral que contextualiza y valora críticamente cada aspecto desarrollado.

En el diagnóstico de la arquitectura actual, se reconoce la necesidad imperante de abordar la situación precaria de la arquitectura, especialmente en relación con los diagramas e implementación. El objetivo era optimizar el flujo y tiempo de extracción de noticias para el desarrollo de futuros reportes. La elección de Jupyter Notebook como herramienta permitió la extracción de noticias y la generación de gráficos, proporcionando una visión general de la arquitectura, identificando defectos y oportunidades de mejora.

En el ámbito de la generación de reportes para el analista, se diseñó un proceso que facilita su uso, sigue un orden coherente y aplica librerías para mejorar la experiencia del usuario. Se espera que futuros analistas encuentren en Jupyter Notebook un entorno fácil de entender y utilizar, con implementaciones que busquen la mejora continua y evolución de los casos de uso.

A pesar de las mejoras implementadas, se reconoce que hubo características y/o mejoras que aún no se implementaron. La idea inicial de un software para generación automática de reportes se ajustó a la complejidad de las variaciones en el formato y presentación. Se descartaron opciones como Power BI debido a limitaciones en la generación de gráficos y la falta de practicidad. La generación automática de reportes sigue siendo un desafío, y se espera que futuras implementaciones aborden la complejidad mediante un enfoque más gradual.

En la fase de implementación de los gráficos generados en el segundo Notebook, inicialmente se buscaba una automatización completa del proceso. Sin embargo, los desafíos surgieron por la naturaleza variable de los gráficos en tamaño y posibles exclusiones según las circunstancias. La decisión de permitir al analista realizar la generación manualmente se tomó para ofrecer flexibilidad y adaptabilidad ante la diversidad de gráficos personalizados. Aunque esto dificulta la posibilidad de que los clientes generen sus propios reportes, abre oportunidades para ajustes específicos según las necesidades de cada caso.

En reflexiones para el futuro, este proyecto ha dejado valiosas lecciones sobre la complejidad de la generación automática de reportes. A medida que se avanza, se contempla la posibilidad de desarrollar soluciones más accesibles para los clientes, quizás explorando tecnologías como Power BI para lograr una interfaz intuitiva. En síntesis, se ha avanzado significativamente en la comprensión y optimización de la arquitectura para la generación de reportes. Las decisiones tomadas ofrecen una base sólida para futuras

mejoras y desarrollos en este campo, invitando a reflexionar sobre cómo estas lecciones pueden aplicarse en proyectos similares, reconociendo la importancia de equilibrar la automatización con la flexibilidad.

8. REFERENCIAS

- Cárcamo, L., Cárdena, C., Vernier, M., Scheihing, E., Sáez, D. (2021-2022),
"Fake news en Chile: ¿Cómo los medios hablan de noticias falsas y desinformación en las redes sociales de Chile?", Proyecto ANID PLU 210013
- Cárcamo, L., Scheihing, E., Vernier, M., Aguilera, O. (2015-2019),
"Redes Sociales y Medios de Comunicación Modelo de análisis basado en minería de datos para la comprensión del ecosistema informativo chileno en internet y la educomunicación ciudadana en la red", Proyecto ANID Fondecyt Regular n°1150545
- Suárez, E., Vernier, M., Huijse, P., Arenas, J., Poblete, V. (2020-2023).
"Sistema integrado de análisis de fuentes sonoras ambientales: Sistema FUSA", Proyecto ANID FONDEF ID20I10333
- Kiran, S. (2022).
2022 Key Trends in Smart Territories. Capgemini.
Accedido el 10 de abril, 2023, desde <https://www.capgemini.com/insights/expert-perspectives/2022-key-trends-in-smart-territories/>
- Ruiz, F. (2020).
Evolution of the gender biases in the news media: a computational method using dynamic topic model. Universidad Austral de Chile.
- Sabol, L. (2023, March 20).
Smart Cities and Democratic Vulnerabilities. NATIONAL ENDOWMENT FOR DEMOCRACY.
Accedido el 20 de abril, 2023, desde <https://www.ned.org/smart-cities-and-democratic-vulnerabilities/>
- Saldias, F. (2021).
Pip install pysophia2: Una librería python para el análisis de medios de prensa internacionales. Universidad Austral de Chile.
- Vásquez, E. (2022).
Recopilación, análisis y visualización de datos de medios de prensa internacionales, periodistas y fuentes de información, Universidad Austral de Chile.
- Vernier, M. (2017),
"Proyecto Voucher de Innovación=> Sistema de aprendizaje de usuario". Corfo (16VIP-71801).
- Vernier, M. (2019-2023),
"Sophia 2.0: Robust methods based on Computational Linguistics and Machine Learning to analyse Media Pluralism", Proyecto ANID Fondecyt Iniciación n° 11190714.

9. ANEXOS

Anexo A: Código del JSON para la búsqueda de documentos.

```
{
  "news": {
    "aliases": {},
    "mappings": {
      "properties": {
        "country": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "date": {
          "type": "date"
        },
        "day": {
          "type": "long"
        },
        "id_news": {
          "type": "long"
        },
        "media_outlet": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "month": {
          "type": "long"
        },
        "text": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "text_length": {
          "type": "long"
        },
        "title": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",

```

```
        "ignore_above": 256
      }
    },
    "url": {
      "type": "text",
      "fields": {
        "keyword": {
          "type": "keyword",
          "ignore_above": 256
        }
      }
    },
    "year": {
      "type": "long"
    }
  }
}
```