Universitat Oberta de Catalunya (UOC)

Máster Universitario en Ciencia de Datos (*Data Science*)

TRABAJO FINAL DE MÁSTER

Área: Data Analysis y Big Data

Sistema de Big Data para Monitorización, Predicción y Análisis de Servicios Web y Móviles para la mejora de la experiencia de usuario

---

Autor: Fiorella Piriz Sapio

Tutor: Rafael Luque Ocaña

Profesor: Albert Solé

Barcelona, 18 de diciembre de 2023

Créditos/Copyright

Una página con la especificación de créditos/copyright para el proyecto (ya sea aplicación por un lado y documentación por el otro, o unificadamente), así como la del uso de marcas, productos o servicios de terceros (incluidos códigos fuente). Si una persona diferente al autor colaboró en el proyecto, tiene que quedar explicitada su identidad y qué hizo.

A continuación se ejemplifica el caso más habitual, aunque se puede modificar por cualquier otra alternativa:

FICHA DEL TRABAJO FINAL

Título del trabajo:	Sistema de Big Data para Monitorización, Predicción y Análisis de Servicios Web y Móviles para la mejora de la experiencia de usuario
Nombre del autor:	Fiorella Piriz Sapio
Nombre del colaborador/a docente:	Albert Solé
Nombre del PRA:	Rafael Luque Ocaña
Fecha de entrega (mm/aaaa):	MM/AAAA
Titulación o programa:	Máster Universitario en Ciencia de Datos (*Data Science*)
Área del Trabajo Final:	Data Analysis y Big Data
Idioma del trabajo:	Español
Palabras clave	Big Data, Análisis Real Time, NLP

Agradecimientos

Quisiera expresar mi profundo agradecimiento a todas las personas e instituciones que han contribuido de manera significativa al desarrollo y éxito de este Trabajo de Fin de Máster.

En primer lugar, quiero agradecer a **NexTReT**, por brindarme la oportunidad de llevar a cabo este proyecto en el contexto de la empresa. Su apoyo continuo, recursos y compromiso con la innovación han sido fundamentales para la realización de esta investigación. La colaboración con los profesionales de la empresa ha enriquecido mi comprensión del tema y ha proporcionado una perspectiva valiosa que ha influido positivamente en el desarrollo de esta tesis.

No puedo pasar por alto el apoyo incondicional de mi **familia**, quienes siempre han estado ahí para brindarme aliento, comprensión y amor. Su respaldo ha sido mi mayor fortaleza y motivación a lo largo de este viaje académico y profesional.

Agradezco también a todos los **profesores, profesoras y tutores** del máster por su dedicación y compromiso. Su trabajo arduo y su pasión por la enseñanza han sido fundamentales para mi formación académica y para la realización de este trabajo. A todos los que han contribuido de alguna manera, mi más sincero agradecimiento. Este logro no habría sido posible sin su colaboración y respaldo.

Resumen

En el mundo actual altamente digitalizado, la **monitorización** eficiente de los servicios web y móviles se ha convertido en un componente crítico para garantizar la **calidad y la experiencia del usuario**. Los servicios y las aplicaciones son cada vez más complejas e interconectadas, es decir, los "frontales"que ven los usuarios en realidad hacen polling de una gran catidad de servicios por detrás.

En los últimos años se ha convertido en una práctica muy común el uso de **robots** de testeo o QA (Quality Assurance), diseñados para simular el comportamiento humano en aplicaciones y entornos, de forma que se puedan detectar errores y se asegure una alta disponibilidad de los datos. En el entorno digital, las validaciones funcionales son importantes porque las latencias cuestan clientes y las indisponibilidades cuestan mucho dinero.

Sin embargo, el mantenimiento de este tipo de sistemas de monitorización no es trivial y requiere una arquitectura de **Big Data** correctamente diseñada y de una herramienta de explotación y visualización de datos que facilite la detección de errores y la generación de alertas.

Por ello en este Trabajo de Fin de Máster se propone el desarrollo de una arquitectura de colas de información mediante el uso de Apache Kafka, cuyos datos son preprocesados de forma simple con Apache Nifi, antes de ser almacenados en ElasticSearch para su indexación y consulta rápida.

Para llevar a cabo un **análisis** profundo de los datos recopilados, se empleará Python y Apache Spark, aprovechando su capacidad para el procesamiento distribuido y el análisis de grandes volúmenes de datos.

Esta combinación de tecnologías proporciona una plataforma robusta para la **detección y predicción** de patrones, la generación de alertas en tiempo real y la predicción de posibles problemas en los servicios web y móviles monitorizados.

Palabras clave: Monitorización, Experiencia de usuario, Big Data, Apche Kafka, Apche Nifi, Elastic Search, Python, Apche Spark, Predicción

Abstract

In today’s highly digitized world, efficient **monitoring** of web and mobile services has become a critical component to ensure **user quality and experience**. Services and applications are increasingly complex and interconnected, meaning that the "front end-süsers see actually poll a large number of services behind the scenes. In recent years, the use of testing or QA (Quality Assurance) robots designed to simulate human behavior in applications and environments has become a common practice to detect errors and ensure high data availability. In the digital environment, functional validations are important because latencies cost customers and unavailability costs a lot of money.

However, maintaining this type of monitoring system is not trivial and requires a properly designed **Big Data** architecture and a data exploitation and visualization tool that facilitates error detection and alert generation.

Therefore, in this Master’s Thesis, we propose the development of an information queue architecture using Apache Kafka, where data is preprocessed simply with Apache Nifi before being stored in ElasticSearch for indexing and quick querying.

To perform in-depth **analysis** of the collected data, Python and Apache Spark are used, taking advantage of their capacity for distributed processing and analysis of large volumes of data.

This combination of technologies provides a robust platform for **pattern detection and prediction**, real-time alert generation, and prediction of possible problems in monitored web and mobile services.

Keywords: Monitoring, Big Data, Apache Kafka, Apache Nifi, Elastic Search, Python, Apache Spark, Prediction.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Descripción general del problema

El **análisis de datos masivos** o Big Data proporciona nuevas oportunidades a la sociedad moderna y desafíos a los científicos de datos. Por un lado, el Big Data ofrece grandes promesas para descubrir patrones y heterogeneidades sutiles que no son posibles de distinguir a simple vista, por lo que se requieren grandes conjunto de datos para poder confirmar la existencia de patrones [45].

Es por ello por lo que en la actualidad el término Big data se ha incluido de forma omnipresente en los entornos digitales, transformando la forma en que las organizaciones recopilan, almacenan, procesan y explotan los datos. Este fenómeno se ha materializado gracias a la creciente cantidad de información generada por diversas fuentes, que abarcan desde transacciones en línea y redes sociales hasta dispositivos conectados y sensores.

Sin embargo, el tamaño masivo de la muestra y la alta dimensionalidad del Big Data introducen **desafíos computacionales y estadísticos** únicos, que incluyen la escalabilidad y la limitación de almacenamiento, la acumulación de ruido, la correlación espuria, la endogeneidad incidental y los errores de medición. Estos desafíos son distintivos y requieren una atención crítica y una estrategia efectiva para su gestión.

Tal y como se menciona en [47], la automatización y la aseguración de calidad (Quality Assurance en inglés) son dos de los grandes retos de los entornos Big Data. Cada vez es más importante asegurarse de que los procesos automatizados estén funcionando correctamente y garantizar que las organizaciones y las personas cuyas vidas son impactadas por sus algoritmos sean tratadas de manera justa.

Por ello el objetivo de este proyecto es asegurar la monitorización de entornos y, sobre todo, de experiencia de usuarios, mediante la creación de una arquitectura que permita minimizar los efectos del:

- Manejo del **volumen masivo** de información que se genera constantemente. La explosión de datos ha superado con creces la capacidad de las infraestructuras de almacenamiento y procesamiento convencionales. Para abordar este desafío, las

organizaciones deben invertir en sistemas escalables y versátiles, así como en tecnologías de almacenamiento y procesamiento distribuido que les permitan manejar la gran cantidad de datos de manera eficiente.

- La **velocidad** a la que se generan los datos también representa un reto importante ya que los flujos de información en tiempo real, desde las transacciones financieras hasta las actualizaciones de redes sociales, requieren sistemas de procesamiento y análisis muy optimizados. La gestión de datos en constante evolución se vuelve esencial para tomar decisiones oportunas y mantenerse competitivo en entornos empresariales dinámicos.
- Los datos pueden ser estructurados, semiestructurados o no estructurados, y provienen de una **variedad** de fuentes y formatos, como texto, imágenes, vídeos y datos de sensores. Integrar y analizar diferentes tipos de datos puede ser complejo y requiere herramientas y técnicas avanzadas de procesamiento y análisis.
- La **veracidad** de los datos es un aspecto crítico, ya que los datos inexactos, incompletos o sesgados pueden llevar a decisiones erróneas. Garantizar la calidad y la precisión de los datos es esencial, lo que implica realizar actividades de limpieza y validación de datos de manera sistemática.
- La capacidad de **extraer valor** de los datos también plantea un desafío. Los conjuntos de datos masivos pueden ocultar información valiosa, y encontrar patrones significativos requiere algoritmos avanzados de análisis de datos y aprendizaje automático. Además, comunicar los hallazgos de manera efectiva a las partes interesadas es esencial para aprovechar al máximo el valor del Big Data.
- La **privacidad y la seguridad** también son preocupaciones cruciales en el mundo del Big Data, especialmente cuando se trata de la gestión de datos personales. La protección de la información sensible y el cumplimiento de regulaciones como el Reglamento General de Protección de Datos (RGPD) son esenciales para evitar riesgos legales y daños a la reputación.
- En los entornos dedicados a la monitorización de servicios, que pueden incluir APIs, sitios web, aplicaciones móviles y otros, se gestionan numerosas validaciones debido a que los frontales recopilan información de múltiples fuentes de datos. La presencia de una gran cantidad de información puede evidenciar errores, pero no necesariamente identifica la **causa subyacente del problema**. Para abordar este desafío, es crucial emplear tecnologías de Big Data que permitan correlacionar la información y descubrir las causas raíces de los problemas.

Para ello se emplearán tecnologías de procesamiento de datos como Apache Kafka, Apache Nifi o Apache Spark, y se realizará un profundo análisis de datos con Python. Estos datos serán almacenados en una base de datos relacional como SQL Server y ElasticSearch[https://elastic.co/es/] para almacenar la información de explotación inmediata. Finalmente, la extracción de estadísticas y de conclusiones será posible mediante el uso de herramientas de visualización como Grafana , Kibana o librerías de React.

1.1.1. Problemas y riesgos

El desarrollo de un proyecto de Big Data utilizando tecnologías Open Source (Apache Hadoop) en un entorno nuevo con migración de datos puede presentar una serie de problemas y riesgos, tales como:

1. Complejidad en la migración de datos: La migración de datos desde sistemas existentes a un entorno de Hadoop puede resultar compleja y propensa a errores. La falta de una estrategia de migración sólida y la incompatibilidad de los formatos de datos pueden generar pérdida de datos y corrupción de la información durante el proceso de transferencia.

2. Escalabilidad y rendimiento: Aunque Hadoop ofrece escalabilidad y rendimiento para grandes volúmenes de datos, su implementación en un entorno nuevo requiere una cuidadosa planificación y configuración. Una infraestructura inadecuada puede resultar en cuellos de botella de rendimiento y tiempos de respuesta lentos, lo que afecta la eficiencia y la usabilidad del sistema.

3. Gestión de la complejidad: El uso de tecnologías de Hadoop implica un entorno complejo que requiere conocimientos especializados para su gestión y mantenimiento. La falta de habilidades y experiencia adecuadas en el equipo puede conducir a errores de configuración y un manejo ineficiente de la infraestructura, lo que afecta la integridad y seguridad de los datos.

4. Costos y presupuesto: La implementación de un proyecto de Big Data basado en Hadoop puede resultar costosa en términos de infraestructura, licencias de software, personal especializado y capacitación. La falta de un presupuesto adecuado y una estimación precisa de los costos puede conducir a desviaciones financieras y a la interrupción del proyecto en fases críticas.

5. Riesgos de seguridad y cumplimiento: La migración de datos a un nuevo entorno de Big Data conlleva riesgos de seguridad, como la exposición de datos sensibles y la falta de medidas de protección adecuadas. Además, el incumplimiento de las regulaciones y normativas de protección de datos puede resultar en sanciones legales y daños a la reputación de la empresa.

Sin embargo, dado nuestro sólido dominio de estas herramientas y nuestra experiencia previa en migraciones similares, aprovecharemos nuestros conocimientos previos y soluciones exitosas para diseñar una arquitectura resistente y adaptable.

1.2. Motivación personal

El desarrollo de un proyecto en el campo del **Big Data**, aplicando técnicas de **Inteligencia Artificial (IA)** y **Aprendizaje Automático (ML)**, como parte de mi Trabajo Final de Máster en ciencia de Datos, representa una oportunidad verdaderamente emocionante y altamente motivadora. Esta combinación de Big Data, IA y ML abre un abanico de posibilidades que profundizan aún más mi entusiasmo por este proyecto.

En primer lugar, la aplicación de técnicas de IA y ML en proyectos de Big Data es un campo en constante evolución que se encuentra en la vanguardia de la **innovación** tecnológica. La posibilidad de trabajar con algoritmos de ML y construir modelos predictivos que puedan aprender y adaptarse a partir de los datos es extremadamente estimulante. Además, Abordar problemas complejos y descubrir patrones ocultos en los datos a través de la inteligencia artificial es un reto técnico que me motiva profundamente y que me ayudará a seguir avanzando en el mundo de la ciencia de datos.

Además, la IA y el ML tienen un **potencial transformador** en una amplia gama de aplicaciones, desde la atención médica y la recomendación de contenido hasta la detección de fraudes y la automatización de procesos empresariales. Contribuir a la aplicación de estas tecnologías para resolver problemas del mundo real y mejorar la eficiencia y la toma de decisiones es una fuente significativa de motivación.

Desde una perspectiva profesional, la demanda de expertos en Data Science, IA y ML sigue creciendo, y liderar un proyecto exitoso que integre estas disciplinas puede abrir puertas a oportunidades profesionales emocionantes y gratificantes en el mercado laboral actual.

En última instancia, este proyecto me dará la posibilidad de avanzar en la **comprensión** de cómo las técnicas de IA y ML para el análisis de Big Data, un entorno en el que llevo algunos años trabajando y formándome, pero en el que todavía me falta mucho por descubrir. Por tanto, mi experiencia me puede aportar una gran ayuda para este proyecto, pero también es un reto para adentrarme más en este sector.

1.3. Objetivos

Los objetivos de este Trabajo de Fin de Máster (TFM) se pueden representar en el siguiente listado:

- Analizar el **estado del arte y las aproximaciones** más comunes para problemas similares al nuestro, es decir, para un entorno de grandes volúmenes de datos con datos a Tiempo Real.
- Diseñar una arquitectura de almacenamiento, análisis y visualización de datos que más se adecue a nuestras necesidades.
- Aplicar **técnicas de Inteligencia Artificial** y Machine Learning para la detección de patrones, análisis de datos temporales, predicción de imágenes o análisis de textos.
- Utilizar herramientas de visualización modernas para extraer conclusiones de los datos analizados.
- Finalmente, un objetivo secundario y menos relacionado con este TFM, es la realización de una aplicación web que permita a las/os técnicas/os e ingenieras/os de monitorización **detectar errores en los entornos y predecir posibles fallos**.

En el día a día de un/a técnico/a o ingeniera/o de Control de Calidad (QA) el poder disponer de un sistema de monitorización avanzado en el que puedan identificar problemas de forma temprana, y sobre todo, el origen del error, les ahorra mucho tiempo y recursos. Además, esto contribuye notablemente a la mejora de la **calidad del servicio ofrecido** y de la eficiencia en el análisis de problemas.

Por lo tanto, el objetivo global de este proyecto es generar una herramienta para las personas involucradas en el Control de Calidad, que les permita detectar problemas de manera temprana, mantener y mejorar la calidad de los sistemas, automatizar tareas de supervisión y garantizar el cumplimiento de los requisitos de calidad y regulaciones, lo que a su vez mejora la experiencia del usuario final y la confiabilidad de los sistemas.

1.3.1. Metodología

En esta sección, se detallan los pasos seguidos para implementar este proyecto y se describen las herramientas utilizadas para hacerlo posible.

Dado que el objetivo principal de este trabajo es desarrollar una herramienta de análisis y predicción de datos de Control de Calidad, se iniciará con un estudio exhaustivo

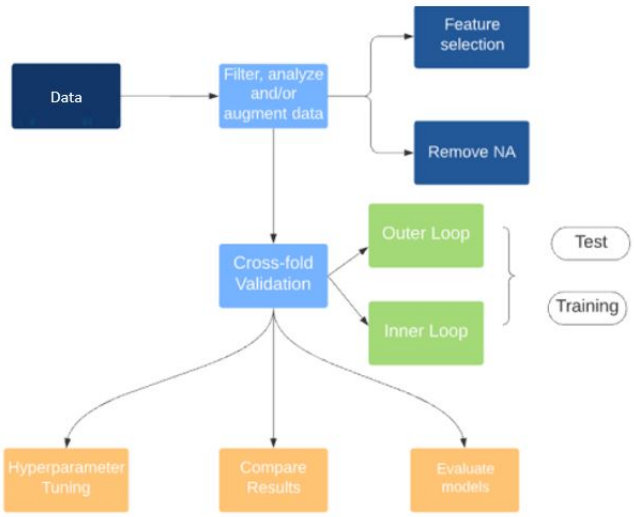

Figura 1.1: Descripción de las fases de desarrollo de los modelos.

del estado actual relacionado con este tipo de entornos y el tratamiento de estos datos. El objetivo será encontrar la mejor manera de **extraer información relevante** que facilite la detección de errores, entre otros objetivos previamente mencionados.

Una vez comprensible el estado del arte, se seleccionará la arquitectura más adecuada para este caso, optando por herramientas de código abierto como Kafka, Nifi y ElasticSearch. Estas herramientas han sido elegidas debido a la experiencia previa y el conocimiento de sus funcionalidades. Se instalarán estos componentes en un clúster de máquinas virtuales proporcionado por Proxmox, diseñando la arquitectura de manera que garantice una alta disponibilidad y la seguridad de los datos.

Con todos los componentes en funcionamiento, se comenzará a recopilar datos desde la cola de Kafka y se preprocesarán mediante Apache Nifi. Estos datos se almacenarán en crudo (RAW) en Elastic, y se diseñará un script en Python para aplicar técnicas de Extracción, Transformación y Carga (ETL), almacenando el resultado en una índice de Elastic.

Una vez que los datos están limpios y preparados para su análisis, se desarrollarán enfoques diferentes según la naturaleza de los datos. Para los datos temporales, se diseñará un modelo de Aprendizaje Profundo para la predicción de series temporales. Para las imágenes, se creará un detector de patrones, y para las páginas web, se aplicarán técnicas de Procesamiento del Lenguaje Natural (NLP).

Todos los modelos implementados pasarán por las siguientes fases 1.1:

- Ajuste de Hiperparámetros: Elegir conjuntos de parámetros para los algoritmos de aprendizaje que ofrecieran el mejor rendimiento del modelo (óptimo).

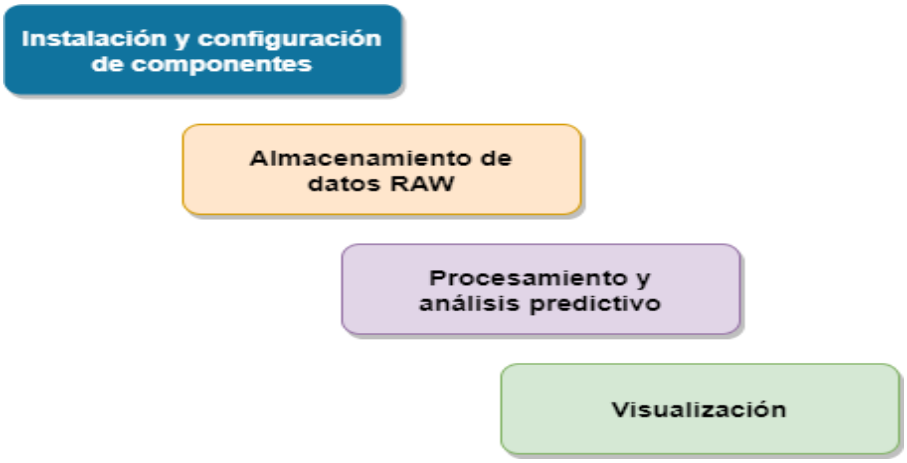

Figura 1.2: Esquema de la metodología implementada.

- Validación Cruzada: Aplicar un procedimiento a los modelos de Aprendizaje Automático para obtener información sobre su capacidad de generalización.
- Comparación de Modelos: Evaluar y comparar diferentes modelos para seleccionar aquel con un rendimiento óptimo según los datos proporcionados.
- Una vez entrenados y validados los modelos, se compararán sus puntuaciones para determinar cuál de ellos ofrece el rendimiento más preciso.

Finalmente, para la visualización de los datos, se utilizarán herramientas como Grafana y Kibana. Además, los resultados de este proyecto se emplearon para crear un sitio web de soporte para los técnicos y profesionales involucrados.

1.4. Los Objetivos de Desarrollo Sostenible (ODS)

Los Objetivo de Desarrollo Sostenible forman parte de la agenda 2030, establecida en 2015 con el fin de proporcionar un plan conjunto para la paz y la prosperidad de las personas y el planeta. Estos 17 objetivos, respaldados por 169 metas específicas, representan un llamada universal a la acción para abordar los desafíos más apremiantes del mundo .

Considerar la influencia de un proyecto en los ODS es esencial para alinear las acciones con principios de responsabilidad social y desarrollo sostenible. Por tanto, emplearemos estos objetivos como una guía para identificar si el impacto social, económico y/o medioambiental de nuestro proyecto contribuye a los esfuerzos globales para abordar desafíos cruciales [**?**].

Figura 1.3: Los 17 Objetivos de Desarrollo Sostenible (ODS), tomada de [16].

A continuación, mencionaremos los principales objetivos que se abordan con el proyecto en cuestión:

- ODS 9 - Industria, Innovación e Infraestructura.

  La implementación de un sistema de Big Data implica avances tecnológicos y contribuye notablemente al desarrollo de infraestructuras robustas para el monitoreo y análisis de servicios digitales [13]. Además, fomenta la innovación y la investigación científica para mejorar los servicios puestos a disposición de la ciudadanía.

- ODS 11 - Ciudades y Comunidades Sostenibles.

  De forma indirecta este proyecto está alineado con las metas del objetivo 11 [5], ya que está herramienta se emplea para la monitorización de servicios de múltiples sectores como bancos, centros sanitarios, aplicaciones públicas de movilidad, seguros de hogar, etc., asegurando que todo el mundo tiene acceso a estos servicios.

- ODS 3 - Salud y Bienestar [17]. Al igual que en aparatado anterior,la capacidad de monitorizar y analizar servicios de salud en línea puede contribuir a la mejora de la atención médica digital, promoviendo la salud y el bienestar.

- ODS 4 - Educación de Calidad. https://www.un.org/sustainabledevelopment/es/education/

  Tanto los servicios que puede ofrecer la herramienta desarrollada como el trabajo de investigación que se ha realizado, pueden contribuir de forma positiva a la formación y al desarrollo académico se otras personas u organizaciones que ofrezcan servicios similares.

- ODS 8 - Trabajo Decente y Crecimiento Económico. https://www.un.org/sustainabledevelopr
  growth/

Toda creación tecnológica requiere de conocimiento y de personal de mantenimiento de los servicios, por lo que servicios como este pueden contribuir al crecimiento económico y a la creación de empleo en el sector tecnológico.

- ODS 16 - Paz, Justicia e Instituciones Sólidas.

  Estas técnicas de seguimiento y control de los servicios web y móviles permiten controlar el acceso a los mismos, de forma que se puedan detectar comportamiento maliciosos sobre todo en sectores como el bancario, dónde cualquier amenaza puede tener una afectación crítica. Por tanto, se contribuye a la seguridad digital y a la prevención de delitos en línea.

- ODS 17 - Alianzas para lograr los objetivos. Poco o casi ninguno de los proyectos tecnológicos se pueden llevar acabo sin el apoyo de la comunidad, que mantiene herramientas de código abierto (como todas las empleadas en este caso) y además aporta conocimiento y soporte para la creación de aplicaciones y servicios tecnológicos.

1.5. Programación

A continuación se presenta la planificación del proyecto:

1.5.1. Auditoría y Revisión

Descripción: En esta fase inicial, se llevarán a cabo revisiones del estado de arte y herramientas similares, y se generará un documento que refleje los objetivos y el alcance del proyecto.

Tarea	Descripción Detallada
Documentación	Revisión de la documentación
Generar un documento del estado actual	Documentar el estado actual del sistema y actualizar los objetivos y el alcance del proyecto.

1.5.2. Análisis Técnico

Descripción: En esta fase, se definirá la arquitectura técnica a desarrollar y se especificará el alcance del proyecto. También se generará un diagrama de componentes y

flujo de datos para visualizar la estructura técnica del sistema.

Tarea	Descripción Detallada
Definir la arquitectura a desarrollar	Establecer la arquitectura técnica que se desarrollará en el proyecto.
Definir el alcance del proyecto	Especificar los límites y alcance del proyecto.
Generar un diagrama de componentes y flujo de datos	Crear representaciones visuales de los componentes y el flujo de datos del sistema.

1.5.3. Prototipo

Descripción: En esta fase, se validarán la arquitectura propuesta y las funcionalidades básicas del sistema a través de la creación de un prototipo.

Tarea	Descripción Detallada
Validar la arquitectura	Realizar pruebas para validar la arquitectura propuesta.
Validar funcionalidades básicas	Verificar el funcionamiento de las funcionalidades esenciales del sistema.

1.5.4. Análisis Funcional

Descripción: En esta fase, se definirá el diseño del dashboard, incluyendo el tipo de gráficas a utilizar, la definición de KPIs y la elaboración del Documento General de Requisitos (DGR).

Tarea	Descripción Detallada
Definición del dashboard	Especificar el tipo de gráficas y elementos que se incluirán en el dashboard.
Definición de KPIs	Identificar los indicadores clave de rendimiento que se medirán en el sistema.
Generar el Documento General de Requisitos	Documentar los requisitos generales del proyecto.

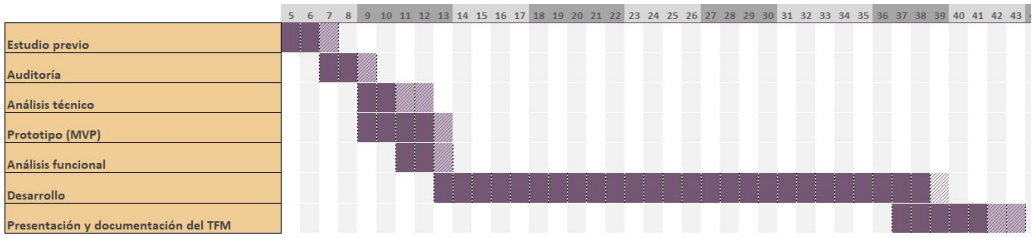

Figura 1.4: Diagrama de Gantt con las fases del proyecto.

1.5.5. Desarrollo

Descripción: En esta fase, se llevará a cabo la instalación y configuración de componentes, el procesamiento de datos antes del almacenamiento, el almacenamiento de datos y el procesamiento posterior al almacenamiento. También se creará la interfaz web y los dashboards.

Tarea	Descripción Detallada
Instalación y configuración de componentes	Configurar los componentes necesarios para el proyecto.
Procesamiento de datos pre-almacenamiento	Realizar el procesamiento de datos antes de su almacenamiento.
Almacenamiento de datos	Almacenar los datos procesados de manera eficiente.
Procesamiento post-almacenamiento	Realizar operaciones adicionales sobre los datos almacenados.
Visualización de datos	Crear una interfaz web y dashboards para la visualización de datos.

1.5.6. Presentación y documentación de la memoria

Descripción: Esta fase se reserva para la documentación exhaustiva y la preparación final de la presentación del proyecto. Aunque la tarea de documentación se llevará a cabo durante toda la duración del proyecto, se prestará una atención especial al final, una vez que todas las tareas estén completas y sea el momento de presentar los resultados obtenidos.

Finalmente, el diagrama de Gantt en el que se presentan de una forma visual todas las fases del proyecto es:

1.6. Resumen de tecnologías empleadas

El la siguiente tabla 5.1 se resumen la principales tecnologías empleadas.

1.7. Resumen de entregables

El código desarrollado, las guies de despliegue de cada componente (en los anexos) y el acceso a la web estática desplegada con GithubPages.

Nombre de la tecnología	Descripción de uso
	Principales paquetes utilizados en Python para aplicar IA
Scikit-learn	Utilizado para crear, entrenar y probar modelos de aprendizaje automático
Tensorflow	"Plataforma de aprendizaje automático de código abierto de extremo a extremo"[7].
Keras	Biblioteca de Python para Aprendizaje Profundo
NLTK	Kit de herramientas de lenguaje natural para procesar datos de lenguaje humano.
	Instrumentos empleados para el desarrollo de aplicaciones web
React	Biblioteca de JavaScript para construir interfaces de usuario en aplicaciones web
MUI	Marco de interfaz de usuario React para construir aplicaciones web receptivas
ApexCharts	Gráficos interactivos de JavaScript para visualizar datos
PM2	Gestor de procesos de producción para aplicaciones Node.js
	Instrumentos empleados para el procesamiento de datos.
Apache Nifi	Plataforma de integración de datos para la automatización del flujo de datos entre sistemas
Apache Kafka	Plataforma de transmisión distribuida para construir canalizaciones de datos en tiempo real
Pandas	Biblioteca de manipulación y análisis de datos para Python
Beautiful Soup	Biblioteca de Python para extraer datos de archivos HTML y XML
requests	Biblioteca de Python para realizar solicitudes HTTP y web scraping
Scipy	Biblioteca científica para matemáticas, ciencia e ingeniería
	Instrumentos empleados para el almacenamiento de datos
Elasticsearch	Motor de búsqueda y análisis distribuido y RESTful
SQL server	Sistema de gestión de bases de datos relacionales
	Herramientas de monitoreo.
Kibana	Herramienta de visualización y exploración de datos para Elasticsearch
Hearthbeat	Agente ligero para monitorear la disponibilidad
Grafana	Plataforma de análisis y monitoreo de código abierto
	Otras herramientas
CentOS	Distribución de Linux que proporciona una plataforma de computación gratuita y de código abierto
HAProxy	Balanceador de cargas
Apache Zookeeper	Coordinación de procesos distribuidos
Visual Studio Code	Editor de código utilizado para crear el proyecto
Gitlab	Gestor de repositorios Git basado en web
Overleaf	Editor de LaTeX basado en la nube
Scopus	Base de datos de resúmenes y citas bibliográficas
Mendeley	Gestor de referencias y red social académica

Cuadro 1.1: Resumen de Tecnologías y Herramientas

Capítulo 2

Estado del arte

Actualmente el sistema de monitorización del entorno de robots está manejado a través de una serie de servicios web (web services) de java y un sql server como base de datos. En el comienzo del proyecto, hace más de 15 años, esta arquitectura era más que suficiente para gestionar los pocos cientos de mensajes que llegaban por hora, sin embargo, a medida que el proyecto fue creciendo, también lo hizo consigo el volumen de datos.

En este aparatado vamos a mencionar las herramientas y aproximaciones existentes, centrándonos en tres características del sistema, la arquitectura, el tipo de datos y la herramienta de visualización.

2.1. Arquitectura

El entorno genera un volumen de datos tan grande que ya no puede ser gestionados y analizados por una herramienta tradicional de procesamiento de datos como SQL Server. De hecho, tal y como se comenta en [23] los sistemas tradicionales de gestión de bases de datos relacionales (RDBMS) fueron diseñados en una época en la que el hardware disponible y los requisitos de almacenamiento y procesamiento eran muy diferentes a los que son hoy en día. Es por ello por lo que este tipo de soluciones han ido encontrado muchos desafíos para cumplir con los requisitos de rendimiento y escalabilidad de esta realidad de **"Big Data"**, siendo cada vez más complejo y desfasado el control de estos sistemas.

Por este motivo han surgido diferentes arquitecturas para mitigar esta problemática. Una de ellas es la computación en la nube, que se asocia con el aprovisionamiento de servicios, en el que los proveedores ofrecen servicios basados en computación a los consumidores a través de la red. A menudo, estos servicios se basan en un modelo de pago por uso donde el consumidor paga solo por los recursos utilizados. En general, un modelo de computación en la nube tiene como objetivo proporcionar beneficios en términos de una menor inversión inicial, costos operativos más bajos, mayor escalabilidad, elasticidad, acceso fácil a través de la Web y reducción de riesgos comerciales y gastos de mantenimiento. Uno de los mayores proveedores de servicios en la nube es Amazon

Web Service [37], una plataforma que proporciona a los desarrolladores y desarrolladoras diversas herramientas necesarias para gestionar y analizar grandes conjuntos de datos de manera eficiente, escalable y segura, lo que les permite extraer información valiosa y tomar decisiones basadas en datos.

De hecho, es interesante notar la cantidad de sinergia entre los requisitos de procesamiento de aplicaciones de Big Data y la disponibilidad y escalabilidad de los recursos computacionales ofrecidos por los servicios en la nube. Sin embargo, aprovechar de manera efectiva la infraestructura en la nube requiere un diseño cuidadoso y la implementación de aplicaciones y sistemas de gestión de datos. Los entornos en la nube imponen nuevos requisitos para la gestión de datos; específicamente, un sistema de gestión de datos en la nube debe tener:

- Escalabilidad y alto rendimiento.
- Flexibilidad según los cambios de las aplicaciones.
- Tolerancia a fallos.
- Seguridad y privacidad.
- Alta Disponibilidad, ya que las aplicaciones no pueden permitirse períodos prolongados de inactividad.

Para ello AWS pone a disposición servicios como las instancias **EC2** (EC2 (Elastic Compute Cloud) [25], que se tratan de máquinas virtuales configurables y redimensionables, que permiten a los usuarios elegir la capacidad de procesamiento, la memoria y el almacenamiento necesarios para ejecutar aplicaciones y cargas de trabajo específicas. Además, EC2 ofrece una amplia selección de tipos de instancias, desde instancias de propósito general hasta instancias optimizadas para cómputo, memoria, almacenamiento o GPU, entre otros, lo que brinda a los usuarios la capacidad de adaptarse a diversas aplicaciones y requisitos de rendimiento.

Otra tecnología que se ha puesto muy de moda en la última década son los son las **plataformas "serverless"**,o "sin servidor", que tal y como mencionan G. McGrath and P. R. Brenner en [48], se trata de *üna oferta en la nube donde la lógica de la aplicación se divide en funciones y se ejecuta en respuesta a eventos".*

El término "sin servidor"no significa que el código no se ejecute en servidores, sino que no es necesario administrar ni preocuparse por los servidores en los que se ejecutan las aplicaciones. De esta forma, se ahorra tiempo de mantenimiento y se aumenta el tiempo dedicado al desarrollo de los productos. Para ello, AWS pone a disposición AWS Lambda [32], que se emplea para tareas sencillas y de corta duración, pues solo se paga por el tiempo de cómputo que consume. Un ejemplo sería la gestión de imágenes en AWS S3, un sistema de almacenamiento de datos en la nube. A continuación se muestra un

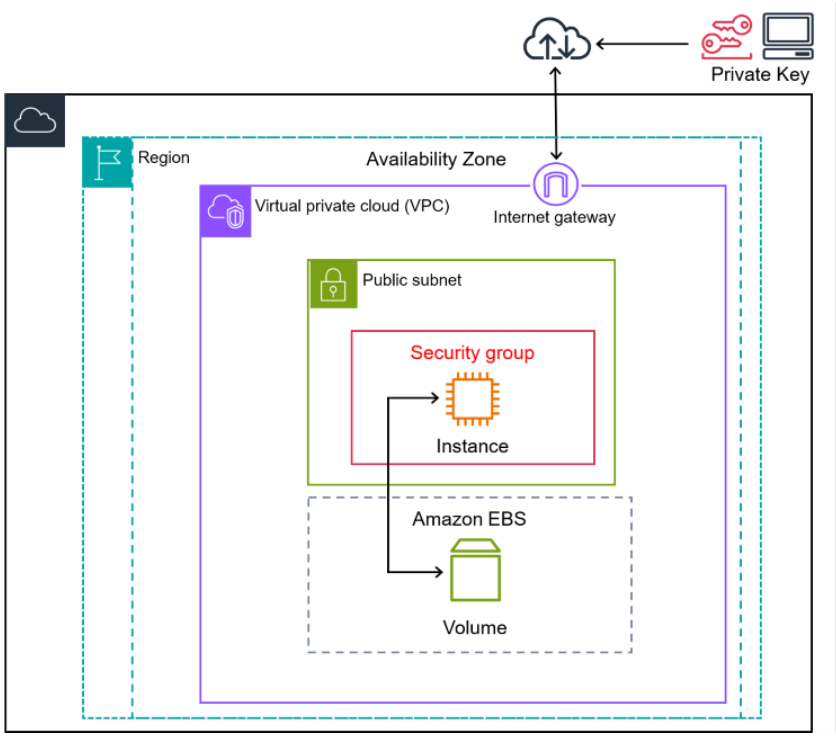

Figura 2.1: Arquitectura básica de Amazon EC2 [27].

ejemplo en la figura 2.2.

Por otra parte, para los casos en los que el entorno requiera de una mayor escalabilidad fiabilidad y seguridad, se propone el uso de contenedores que permiten añadir instancias de un componente de una forma muy sencilla y flexible, es decir, se pueden escalar cuando la carga de trabajo sea mayor y eliminar cuando esta disminuya. Para ello, AWS dispone de Amazon Elastic Container Service (ECS) [30] y Amazon Elastic Kubernetes Service (Amazon EKS) [31]. Ambos son servicios basados en la gestión de contenedores (Docker principalmente), aunque EKS proporciona un entorno de Kubernetes totalmente administrado.

Estas son algunas de las herremientas propuestas por AWS para la gestión de entornos big data, aunque existen otros proveedores con una gran comunidad como son Google Cloud Platform o Microsoft Azure, que proponen servicios similares.

A pesar del gran potencial de estas plataformas, tienen en común una serie de características que hacen que no sean la mejor implementación para nuestro proyecto. Algunas de ellas son:

- Costes: El sistema de precio se basa en el pago por tiempo de de cómputo utilizado. Esto puede aumentar significativamente si no se gestionan adecuadamente los recursos en la nube. Además, es muy recomendables en los casos en los que no se

Figura 2.2: Procesamiento de imágenes con AWS Lambda [24].

disponga de una infraestructura IT.

- Conectividad a Internet: La falta de conectividad puede provocar interrupciones en el servicio y afectar el rendimiento de las aplicaciones críticas para el negocio.
- Seguridad y privacidad: Las violaciones de seguridad, la pérdida de datos y el acceso no autorizado son riesgos potenciales que deben abordarse con medidas de seguridad y políticas robustas.
- Integración y migración: Desarrollar una plataforma en la nube suele implicar la migración del entrono anterior a la nube para facilitar el mantenimiento y la integración, cosa que llevaría un gran esfuerzo en nuestro caso.

Por todos los aspectos comentados, hemos optado por una plataforma de código abierto como es Apache Hadoop. En concreto se trata de un enfoque moderno para el análisis de datos no estructurados (principalmente) y que ofrece una solución potente y flexible para el procesamiento de grandes volúmenes de datos. Está basado en la tecnología **MapReduce**, permitiendo así el procesamiento distribuido de conjuntos de datos masivos en clústeres de servidores, lo que facilita la escalabilidad y el procesamiento eficiente de los datos. Además del paradigma MapReduce, Hadoop también incluye el sistema de archivos distribuido **Hadoop (HDFS)**, que permite el almacenamiento distribuido de datos en múltiples nodos de un clúster. Esta arquitectura descentralizada y su capacidad para manejar datos no estructurados lo convierten en una opción atractiva para empresas que necesitan procesar y analizar grandes volúmenes de datos de diversas fuentes de manera eficaz y rentable [7].

Además de MapReduce y HDFS, Hadoop pone a disposición componentes como **Apache Hive**, una herramienta que proporciona una interfaz de consulta similar a SQL para

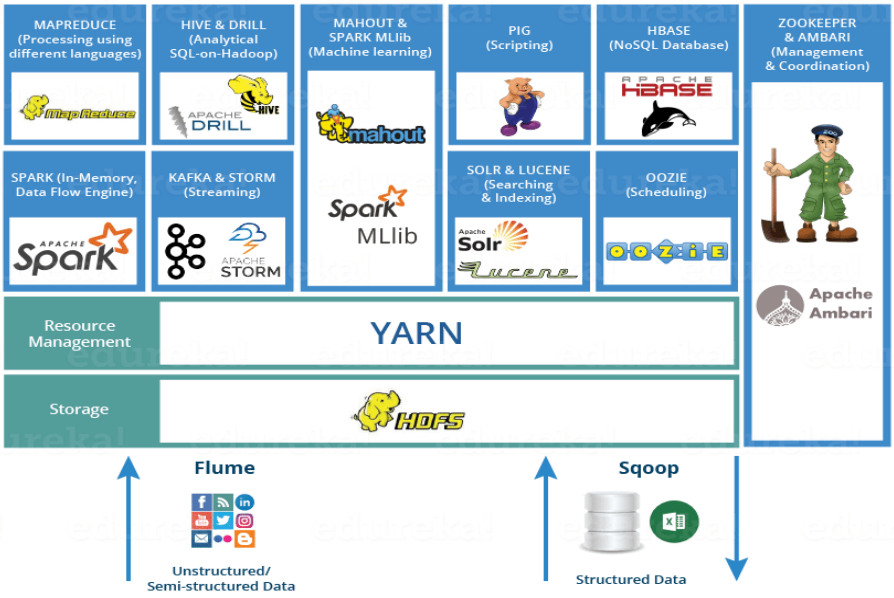

Figura 2.3: Ecosistema de Apache Hadoop [29].

analizar y procesar datos almacenados en Hadoop; **Apache Pig** para el diseño de programas que ejecutan tareas complejas; Apache Hbase como base de datos NoSQL que es ideal para casos de uso en los que se requiera un acceso aleatorio de los datos y en tiempo real; **Apache Spark**, que está muy bien integrado en el ecosistema Hadoop y que proporciona un procesamiento de datos en memoria rápido y eficiente, ofreciendo capacidades de procesamiento de datos en tiempo real y análisis de datos complejos; para la orquestación de flujos de trabajo también existe Apache Oozie o Apache Hadoop. Finalmente para el manejo de colas disponemos de Apache Kafka o Apache Flume, tal y como vemos en la siguiente imagen 2.3.

Estas herramientas de código abierto brindan a los usuarios la capacidad de almacenar, procesar y analizar grandes volúmenes de datos de manera eficiente, permitiendo una mayor flexibilidad y capacidad de escalar las operaciones de análisis de datos en entornos distribuidos.

2.2. Tipo de datos

En este proyecto hemos optado por emplear un datos sin estructura fija, dejando atrás el sistema relacional con el fin de adaptarnos a un entorno empresarial en constante evolución. Al hacer esta transición, estamos apostando por la flexibilidad y la escalabilidad que ofrece el almacenamiento de datos no estructurados[39], lo que nos permite gestionar volúmenes masivos de información de manera eficiente. Además, esta migración nos

capacita para realizar análisis más profundos y detallados de conjuntos de datos complejos, lo que amplía nuestra capacidad de tomar decisiones estratégicas informadas. Al ser más ágiles y adaptables, podemos responder con mayor rapidez a las demandas del mercado y explorar nuevas oportunidades de innovación y crecimiento empresarial.

Ante los desafíos que enfrentan los RDBMS tradicionales al manejar Big Data y al satisfacer los requisitos de la nube descritos anteriormente, en los últimos años han surgido una serie de soluciones especializadas en un intento de abordar estas preocupaciones. Las llamadas bases de datos NoSQL y NewSQL se presentan como alternativas de procesamiento de datos que pueden manejar este enorme volumen de datos y proporcionar la escalabilidad requerida.

Tal y como comenta Bakshi, Kapil [7], a diferencia de las esquemas de datos relacionales, las bases de datos NoSQL se centran en el almacenamiento de datos de alto rendimiento y escalabilidad, permitiendo un acceso de bajo nivel a la capa de gestión de datos. El esquema flexible o también denominado **"schemaless"**, facilita las actualizaciones de la estructura de datos sin reescribir tablas, y deja la responsabilidad del mantenimiento del esquema en manos de las aplicaciones que lo consultan.

La inmensa cantidad de soluciones NoSQL existentes y las discrepancias entre ellas dificultan la formulación de una perspectiva en el campo e incluso plantean más desafíos para seleccionar la solución adecuada para nuestro problema en particular. Por ello, debemos estudiar los distintos tipos de modelos para poder elegir el que mejor se adecue a nuestro problema.

Tomando la lista mencionada en [44], podemos diferenciar:

- Los almacenes **clave-valor** tienen un modelo de datos simple basado en pares clave-valor. Son eficientes para el almacenamiento distribuido pero no son ideales para escenarios que requieren relaciones o estructuras complejas.
- Los modelos de **familias de columnas** están inspirados en Google Bigtable y almacenan datos de manera orientada a columnas. Ofrecen una indexación y consulta más potente que los almacenes clave-valor debido a la estructura de columnas y familias de columnas.
- Los **almacenes de documentos** utilizan claves para ubicar documentos y permiten la indexación y consulta basadas en el contenido de los documentos. Son adecuados para aplicaciones que pueden representarse en formato de documento y no requieren un esquema fijo.
- Finalmente, los modelos orientados a grafos utilizan grafos como su modelo de datos y son eficientes en el almacenamiento y recorrido de relaciones entre diferentes

entidades. Son útiles en escenarios que involucran datos altamente interconectados, como redes sociales y sistemas de recomendación.

Por lo tanto, podríamos decir que el esquema de datos que mejor se adecúa a nuestro conjunto es el documental. Por ello emplearemos ElasticSeach como un motor de búsqueda y análisis distribuido que almacena datos en forma de documentos JSON (JavaScript Object Notation) o en un formato derivado de JSON. Como veremos más adelante, los documentos en Elasticsearch son almacenados en índices y cada campo dentro de un documento puede contener diferentes tipos de datos. Este enfoque permite una gran flexibilidad en la estructura de datos, ya que no se requiere que todos los documentos en un índice sigan el mismo esquema. Además, Elasticsearch proporciona capacidades avanzadas de indexación y búsqueda de datos no estructurados o semi-estructurados, lo que lo hace ideal para escenarios en los que se necesita realizar búsquedas y análisis complejos en grandes volúmenes de datos.

2.3. Visualización

Si nos centramos en la representación de los datos, existen diversas tecnologías altamente empleadas en la actualidad que permiten la creación de sashboards de una forma sencilla.

Una de ellas es Grafana [35], una plataforma de código abierto para la visualización y monitorización de métricas, ideal para visualizar datos de series temporales. Es fácilmente integrable con una amplia gama de fuentes de datos, incluidas bases de datos de series temporales, como Graphite, InfluxDB y Prometheus. Además, permite crear dashboards personalizados con una variedad de opciones de visualización, como gráficos de líneas, gráficos de barras y tablas, lo que facilita el seguimiento y análisis de métricas de sistemas y aplicaciones [40].

Otra herramienta que es muy útil para la visualización de de datos y que además está muy bien integrada con ElascticSearch, es Kibana una herramienta de visualización de datos de código abierto ue permite a los usuarios visualizar datos almacenados en Elasticsearch e interpretarlos a través de gráficos, tablas y mapas interactivos [38].

Kibana es popular entre los profesionales de la tecnología de la información y los analistas de datos para analizar y representar datos en tiempo real de manera efectiva.

Finalmente, queremos mencionar Splunk, una plataforma de pago fundada en 2003 con el objetivo de dar sentido a la información de la máquina que contiene muchos datos valiosos que pueden impulsar la eficiencia, productividad y visibilidad para el negocio, tal y como se menciona en [36].

Splunk es conocido por su capacidad para indexar datos de diferentes fuentes y ofrecer capacidades de búsqueda y visualización avanzadas. Permite a los usuarios monitorear el rendimiento de sistemas, aplicaciones y redes, y ofrece paneles personalizables y gráficos interactivos para analizar datos de manera eficiente.

Si bien estas aproximaciones son potentes para la visualización de datos y el monitoreo, pueden presentar desafíos en términos de complejidad de aprendizaje, limitaciones de personalización, costos y dependencia de terceros. En contraste, una aplicación personalizada desarrollada en React o tecnologías similares brinda un mayor control sobre el diseño, la funcionalidad y la escalabilidad del producto, aunque implica una inversión significativa en términos de tiempo, recursos y experiencia en desarrollo. Además, la popularidad de React y su comunidad de desarrollo activa hacen que sea más sencillo encontrar personal con experiencia y conocimientos en el "framework de desarrollo", lo que facilita la construcción, mantenimiento y escalabilidad de la aplicación a medida.

2.4. Análisis de html (NLP)

Con la proliferación de contenido web, hay una creciente demanda de herramientas automatizadas que puedan analizar páginas web y extraer información valiosa. Sin embargo, el desafío radica en la presencia de considerable ruido, como anuncios, barras de navegación y enlaces, que rodea el contenido informativo en las páginas web típicas. Este ruido complica el proceso de extracción de información, lo que hace necesario identificar y aislar el contenido principal para un análisis preciso.

Tradicionalmente, muchos enfoques han dependido de plantillas predefinidas de sitios web y reglas elaboradas a medida para sitios web específicos para extraer contenido relevante. Sin embargo, estos métodos a menudo carecen de adaptabilidad y pueden tener dificultades con la naturaleza dinámica del contenido web. [46]propone un enfoque más genérico que no requiere conocimiento previo de las plantillas de sitios web. En cambio, aboga por una combinación de análisis del Modelo de Objeto del Documento (DOM) de HTML y técnicas de Procesamiento del Lenguaje Natural (PLN).

También se han desarroolado otros proyecto comp *DOM-LM: Learning Generalizable Representations for HTML Documents* [41] o UNDERSTANDING HTML WITH LARGE LANGUAGE MODELS [43] que hacen empleo de Language Learning Models (LLM) con el fin de mejorar la precisión en la extracción de contenido y extraer información relevante.

En este caso, se emplearan técnicas de análisis de texto como los word embeddings, lemantización o tokenización y además se aplicarán modelos de Machine Learning para clasificar los textos obtenidos en categorías.

Capítulo 3

Arquitectura del sistema

Tal y como se introducía en 2.1, para el caso de uso en cuestión, es necesario diseñar una arquitectura de Big Data resiliente y con alta disponibilidad para abordar los desafíos específicos de un entorno en el que se reciben miles de registros por segundo y donde la indisponibilidad del sistema no es una opción. El sistema no solo recibirá operaciones de escritura, sino que será constantemente consultado desde la aplicación que analiza y explota la información recibida.

Por lo tanto, se debe priorizar con la misma importancia las operaciones de lectura y de escritura, lo que se traduce en un sistema consistente y disponible al mismo tiempo. Es decir si tenemos en cuenta el teorema CAP [4] (profundizado en la asignatura de Arquitectura de Bases de Datos no tradicionales), nuestro gestor de almacenamiento debe ser CP (consistencia fuerte), para que no se muestren datos incorrectos, pero al mismo tiempo replicado para asegurar la disponibilidad.

Teniendo en cuenta estos aspectos, se ha diseñado una arquitectura **altamente disponible** mediante la distribución de carga en un **sistema distribuido** con tres nodos, la **replicación** de datos tres veces y la **monitorización** continua para prevenir fallos.

Se prioriza un equilibrio entre operaciones de escritura y lectura mediante técnicas de optimización de bases de datos, sistemas de cola y buffering. Además, se implementa un sistema de **balanceo de cargas** y la **escalabilidad horizontal** para manejar eficientemente el tráfico variable y poder escalar los recursos cuando sea necesario.

A modo resumen:

- Partimos de un sistema actual desfasado donde los datos son almacenados en una base de datos relacional (**SQL Server**) en la que se almacenan valores precalculados resultantes de un proceso de agregación que se ejecuta cada una hora.
- Debido a la poca eficiencia, escalabilidad y velocidad del proceso, se propone una migración en la que se emplearán tecnologías de procesamiento de datos como **Apache Kafka**, **Apache Nifi** o **Apache Spark**, y se realizará un profundo análisis de datos con **Python**.
- Estos datos serán almacenados en una base de datos relacional como **SQL Server**, donde se almacenaran algunas tablas maestras ya existente que son difícil-

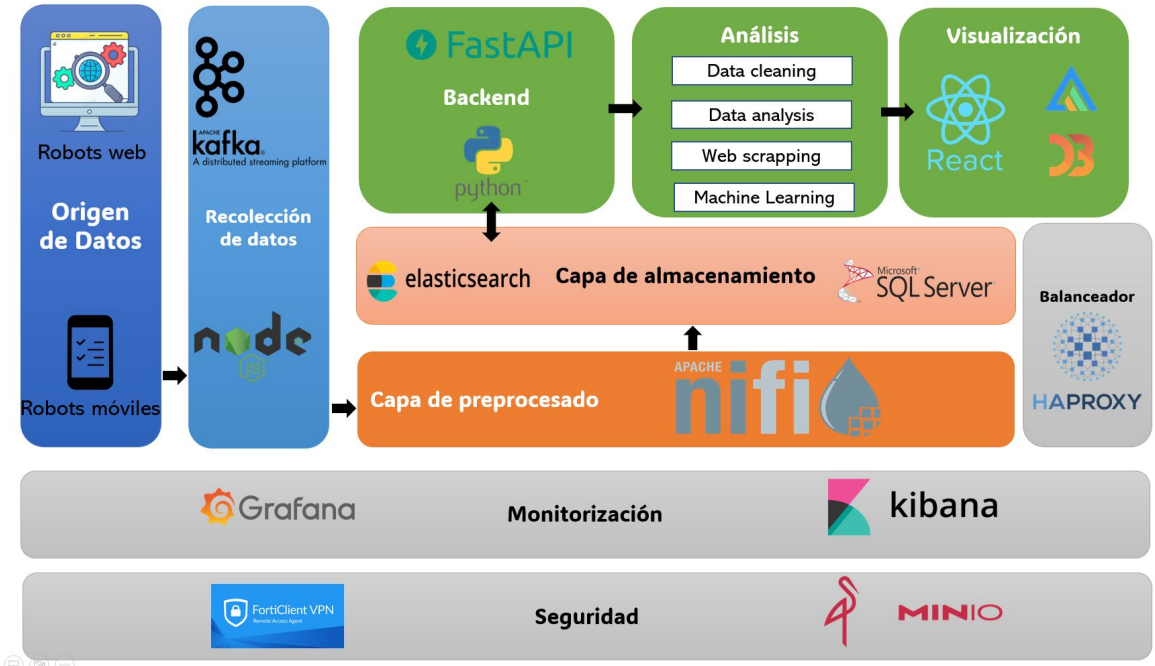

Figura 3.1: Arquitectura del sistema.

mente migrables, y **ElasticSearch** como base de datos NoSQL para almacenar la información de explotación inmediata. Finalmente, la extracción de estadísticas y conclusiones será posible mediante el uso de herramientas de visualización como **Grafana**, **Kibana** o bibliotecas de **React**.

Para la implementación del sistema completo se ha propuesto una arquitectura con las siguientes características 3.1:

- 3 servidores virtuales, gestionados con Proxmox, con las siguientes características:
  - 8 cores
  - 16 GB de RAM
  - 64 GB de disco duro
- 2 servidores para balancear las cargas con instalado y configurado.
- **ElasticSearch**
  - Se ha desplegado un clúster ElasticSearch versión 8.9 con los roles de master y data nodes activado en los tres nodos.

- Para optimizar los accesos todos los índices, estos se configuran siempre con 1 replica y 3 shards para que todas las consultas y accesos se distribuyan en todos los nodos.
- Creación de índices:
  - Se ha creado un índice para los datos RAW, que almacenara toda la información preprocesada desde Nifi.
  - Se ha creado un índice de agregados con la información calculada desde Python.
- Se ha configurado una política de rotado y limpieza de índices de forma que el volumen de datos almacenado en cada índice sea óptimo (máximo 50 Millones de documentos).

- **Kibana**
  - Se ha desplegado Kibana en los tres nodos para disponer de alta disponibilidad en el acceso a los dashboards.
- **Apache Nifi**
  - Se ha configurado un cluster de Nifi versión 1.20.0 con 3 nodos, donde se crearán distintos grupos para generar los flujos de transformación e integración de datos.
  - Existe un Zookeeper para la conexión del cluster.
- **Servidor Node.JS 18.10**
  - Se ha empleado **pm2** para levantar la API de Kafka en Node.JS como servicio en los tres servidores. Además, también se han levantado con pm2 scripts de python que actúan como servicios API.
- **Kafka 3.4.0**
  - Se ha desplegado un clúster de Kafka con 3 brokers para la orquestación de las colas.
  - Existe un Zookeeper para la conexión del clúster.
- **React y FastAPI para la web**

- Se ha creado un a aplicación web balanceada y diseñada con React 17.0.1. Se han empleado librerías como MUI (Material UI) para los estilos, ApexCharts para las gráficas y ReactTables para las tablas filtrables.
- El backend se ha programado con Python 3.11.0, generando una API REST con FastAPI y múltiples servicios que se ejecutan constantemente para generar datos agregados.
- Para disponer de alta disponibilidad para la API y la web de React se montará un balanceo basado en el uso de **HAproxy / keepalived**.

https://online.hbs.edu/blog/post/data-life-cycle

Como hemos visto en asignaturas como Fundamentos de la Ciencia de Datos y Tipología y Ciclo de Vida de los Datos, la importancia de gestionar datos va más allá de la extracción y análisis. La **monitorización** activa juega un papel fundamental para detectar anomalías y prevenir posibles problemas en tiempo real. La implementación de **sistemas de respaldo (backups)** es esencial para garantizar la integridad y disponibilidad de los datos en caso de fallas o pérdidas inesperadas.

Además, se reconoce la necesidad de establecer medidas de seguridad adecuadas, personalizadas según los requisitos específicos del problema abordado. La seguridad de los datos se convierte en un elemento crucial para proteger la privacidad, cumplir con regulaciones y evitar posibles brechas de seguridad. En este contexto, se subraya la importancia de considerar **la confidencialidad, integridad y disponibilidad** de los datos como principios fundamentales en el diseño y la implementación de sistemas relacionados con la gestión de datos. La combinación de monitoreo proactivo, sistemas de respaldo robustos y medidas de seguridad sólidas contribuye a crear un entorno integral y confiable para la gestión de datos en cualquier proyecto.

Por ello además de los servicios comentados anteriormente, se ha añadido:

- Capa de **seguridad** con FortiClient VPN. Además, las contraseñas de los usuarios se han cifrado con claves fernet y se almacenan en la web medianet cookies.
- Capa de **monitorización** con Grafana y Hearthbeat para enviar sondas a los componenetes desplegados y poder establecer alertas si la disponibilidad del cluster se ve afectada.
- Capa de **backups**. Se ha dispuesto se un servidor para desplegar un Minio, servicio de almacenamiento distribuido compatible con AWS S3 y que permite almacenar grandes conjuntos de como backups programados que se generan desde ElasticSearch.

Capítulo 4

Instalación y configuración

En este aparatado se detallarán los procesos seguidos para instalar cada uno de los componentes del cluster. En primer lugar se hará una breve descripción de cada componente y después se comentarán los aspectos principales de la configuración realizada. La información completa sobre el despliegue se dejará en los anexos de este documento.

4.0.1. Elasticsearch

Elasticsearch es un motor de búsqueda y análisis distribuido de código abierto construido sobre Apache Lucene y desarrollado en Java. Originalmente diseñado como una versión escalable del marco de búsqueda Lucene, ha evolucionado hasta convertirse en el componente central del ecosistema llamado "Elastic Stack" [8]. Elasticsearch cumple diversas funciones, actuando como un **motor de búsqueda**, una **base de datos de análisis** y una **solución de big data**, manejando tareas desde búsquedas simples en sitios web hasta análisis de datos complejos y visualización para inteligencia empresarial.

En su núcleo, Elasticsearch funciona como un servidor que procesa solicitudes JSON y devuelve datos en formato JSON. Almacena, busca y analiza grandes volúmenes de datos de manera rápida y casi en tiempo real, proporcionando respuestas rápidas mediante la indexación de documentos en lugar de buscar directamente en el texto. La estructura de Elasticsearch se basa en documentos en lugar de tablas, utilizando llamadas API REST para el almacenamiento y la recuperación de datos[10].

- El motor organiza datos en **documentos** basados en JSON. Funcionan como filas en una base de datos, representando entidades específicas que pueden ser más que solo texto, incluyendo datos estructurados como números y fechas. Cada documento tiene un ID único y un tipo de datos, aunque el esquema es flexible y se puede modificar dinámicamente.

- Estos documentos se almacenan en **índices**, colecciones de documentos con características similares y lógicamente relacionados. Se emplean para realizar operaciones como búsqueda, indexación y/o eliminación de documentos.

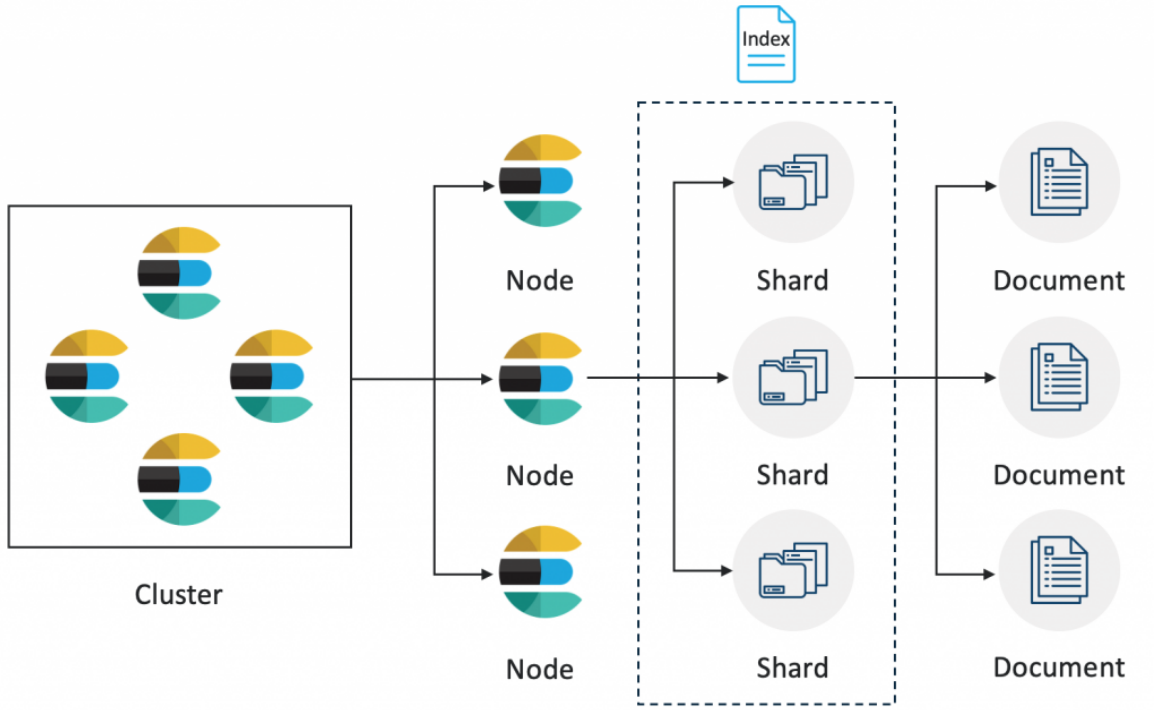

Figura 4.1: Arquitectura del ElasticSearch tomada de [9]

- Además, Elasticsearch utiliza **índices invertidos** que mapean palabras a sus ubicaciones en documentos. Divide documentos en términos de búsqueda y mapea cada término a los documentos que lo contienen. Utilizando índices invertidos distribuidos, Elasticsearch encuentra rápidamente coincidencias para búsquedas de texto completo en grandes conjuntos de datos.

En general, la **versatilidad y escalabilidad de Elasticsearch** lo convierten en una elección popular para una amplia gama de aplicaciones, contribuyendo a su estatus como uno de los motores de búsqueda empresariales más populares y uno de los 10 principales sistemas de gestión de bases de datos.

En nuestro caso, hemos tenido que instalar Elasticsearch en los tres nodos del cluster y modificar el fichero de configuración *elasticsearch.yaml*.

Primero debemos indicar el dns o ip del host en el que se va a levantar la instancia y el dns del resto de instancias que van a formar parte del cluster. Además, indicamos la ruta donde queremos almacenar los datos, solo en caso de que sea distinta a la original.

```
1 # Establecer un nombre al cluster (debe ser el mismo en todos los nodos)
2 cluster.name: cluster-name
3
4 # Establcer un nombre al nodo (debe ser descriptivo)
5 node.name: node-1
```

```
6
7  # Modificar con el nombre del host específico si se
8  #quiere acceder por un nombre que no sea localhost
9  network.host: node-1
10
11 # Modificar con el nombre de todos los host del cluster
12 discovery.seed_hosts: ["node-1", "node-2", "node-3"]
13
14 # Añadir la ruta en la que se quieren guardar los datos
15 path.data: /data/elasticsearch
```

Siguiendo con las guías de buenas prácticas, se ha activado la comunicación por SSL, una vez generados los certificados.

```
1 # Habilitar comunicación por SSL
2 xpack.security.transport.ssl:
3   enabled: true
4   verification_mode: certificate
5   client_authentication: required
6   keystore.path: elastic-certificates.p12
7   truststore.path: elastic-certificates.p12
```

Con esta configuración y siguiendo los pasos de 7.1.1, una vez iniciado el servicio en todos los nodos, podremos acceder a la API y realizar consultas.

Por otra parte, hemos creado los índices usando **“index templates“**, que son plantillas que especifican la configuración y el mapeo de campos para la creación automática de índices. Estos templates permiten simplificar la administración, mejorar el rendimiento y mantener la coherencia en entornos con múltiples índices.

En términos generales, en el template se ha configurado la política de rotado de los índices 4.2, es decir, cada cuanto tiempo o tamaño del índice se va a generar un nuevo índice. Esto es muy importante ya que Elasticsearch no está optimizada para almacenar grandes índices, si no múltiples índices con información relacionada, normalmente por la fecha de introducción. De esta forma, todos los índices se asocian a un “alias“ sobre el que se realizan las llamadas y el motor se encarga internamente de realizar la búsquedas en el índice concreto.

En el template también se suelen indicar los campos y tipos que queremos especificar, pues si no lo hacemos, Elasticsearch los creará con el tipo que detecte por defecto.

A continuación se detallan los índices que se han creado y el formato de los campos que almacenará inicialmente, aunque el esquema es flexible y puede cambiar en cualquier momento.

- **raw-asm**

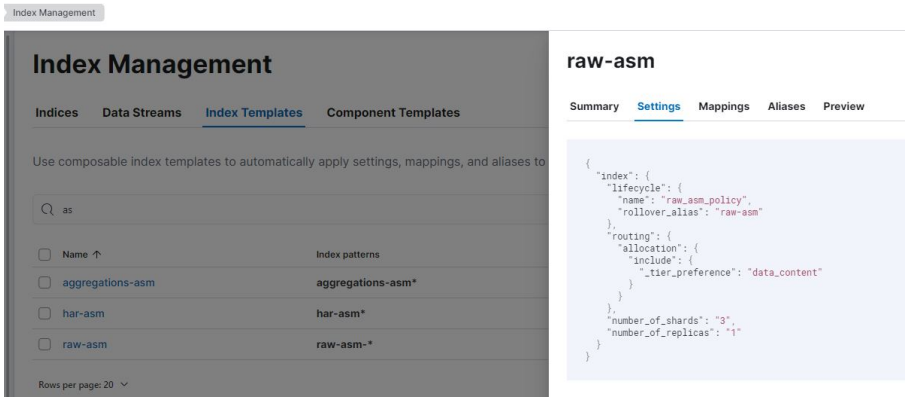

Figura 4.2: Ejemplo de index template.

Ejecuciones de los test con la información preprocesada desde Nifi. Tenderemos ejecuciones por cada uno de los pasos del test (**type=Result**) y ejecuciones agrupadas (**type=GroupedResult**) que incluyen el resultado global del test, unificando todos los pasos.

```
{
   "type":"Result o Groupedesult",
   "nombrePaso":"nombre del paso",
   "idTest":"identificador del test",
   "tiempoPaso":"duracion del paso o del test completo",
   "nombreTest":"nombre identificativo del test",
   "idGrupTest":"identificador del grupo del test",
   "idPaso":"identificador del paso",
   "nombreRobot":"nombre identificativo del robot",
   "fecha":"fecha orginal en formato yyyyMMddHHmmss",
   "fechaProcesado":"fecha transformada en formato yyyy-MM-dd HH:mm:ss",
   "idRobot":"identificador del robot",
   "id":" identificador único de cada registro formado por la fecha, el
      idPaso, el idTest y el idRobot",
   "elementError": "elemento que causa el error, en caso de que lo haya",
   "errorMsg": "mensaje de error que devuelve el test, en caso de que lo
      haya",
   "errorSel": "error que devuelve Selenium, en caso de que lo haya",
   "harFile": "nombre del har, en caso de que exista",
   "idError": "identificador del erorr, en caso de que exista",
   "pathError": "ruta al fichero de error, en caso de que exista",
   "urlFailed": "url en la que se origina el error del test, en caso de que
```

```
      exista"
}
```

- **aggregations-asm**

  Contiene información sobre los cálculos calculados para los test agrupados por el rango de fechas indicadas en el filtro, por defecto se filtra por día, es decir desde la hora de consulta hasta 24 horas menos. Se emplea como chaché de búsquedas para la web pero luego en la web se pueden hacer búsquedas por otros tramos horarios, cálculos que se hacen a tiempo real.

  Para que esta información se actualice cada minuto, se ha añadido un servicio que realiza el cálculo cada minuto.

```
{
   "idTest":"identificador del test",
   "porcentajeCorrecto":"porcentaje de ejecuciones correctas agrupadas por
      la fecha de consulta (por defecto por día)",
   "totalCorrecto":"número de ejecuciones correctas",
   "totalError":"número de ejecuciones erróneas",
   "total":"número total de ejecuciones",
   "fechaCalculo":"fecha de cálculo en formato yyyy-MM-dd HH:mm:ss,
   "activo":"true o false dependiendo si hay ejecuciones del test en la
      fecha indicada",
   "fechaUltimaEjecucion":"fecha de la última ejecución del test en formato
      yyyy-MM-dd HH:mm:ss",
   "duracionUltimaEjecucion":"duración de la última ejecución del test",
   "tags": "lista de tags asociados a un test",
   "entorno":"nombre del entorno al que pertenece el test",
   "nombreTest":"nombre del test",
   "nombreRobot":"nombre del robot",
   "errorUltimaEjecucion": "true o false dependiendo si el test falló en la
      última ejecución"
}
```

- **har-asm** Este índice se emplea para almacenar los **HAR (HTTP Archive)** de los test que traen esa información. En concreto los fichero ".har" se tratan de un registro de la interacción de un navegador web con un sitio, informando sobre los detalles sobre las solicitudes HTTP, respuestas, tiempos de carga de recursos, cookies, y más información. En nuestro caso es interesante almacenar esta información

porque nos puede dar pistas del punto en el que se produjo el error del test y el motivo, además de otros problemas que pueda haber en los servidores. Por ello almacenamos esta información para procesarla posteriormente.

```
{
   "id":"identificador único de cada registro formado por la fecha, el
      idPaso, el idTest y el idRobot",
   "idPaso":"identificador del paso",
   "idRobot":"nombre del robot",
   "idTest":"identificador del test",
   "type": "HAR",
   "fecha":"fecha orginal en formato yyyyMMddHHmmss",
   "fechaProcesado":"fecha transformada en formato yyyy-MM-dd HH:mm:ss",
   "HARName": "nombre del HAR",
   "HARFile": "JSON con el contenido del HAR"
}
```

4.0.2. Kibana

Kibana es otro de los componentes del “Elastic Stack“, que proporciona una herramienta de interfaz visual para explorar, visualizar y construir dashboards sobre los datos almacenados en Elasticsearch [22].

La función principal de Kibana es la consulta y análisis de datos. Además, las características de visualización de Kibana te permiten representar los datos de diversas maneras mediante mapas de calor, gráficos de líneas, histogramas, gráficos circulares y soporte geoespacial.

En este caso se ha empleado principalmente para crear dashboards de monitorización de los índices es decir, saber cuántos datos están llegando y qué tipo de información se recibe en cada momento.

Asimismo, se ha configurado el HearthBeat [11], un sensor ligero que permite verificar periódicamente el estado de los servicios y determinar si están disponibles. Este componente proporciona una forma eficiente de controlar la disponibilidad de servicios al realizar verificaciones regulares, asegurando así que los sistemas estén operativos y respondan como se espera. La forma visual de manejar la información generada por el HearthBeat es mediante la creación de dashboards en Kibana o Grafana, pero en este se caso se ha optado por la primera opción por simplicidad.

Los principales cambios realizados en el fichero de configuración *kibana.yaml* han sido:

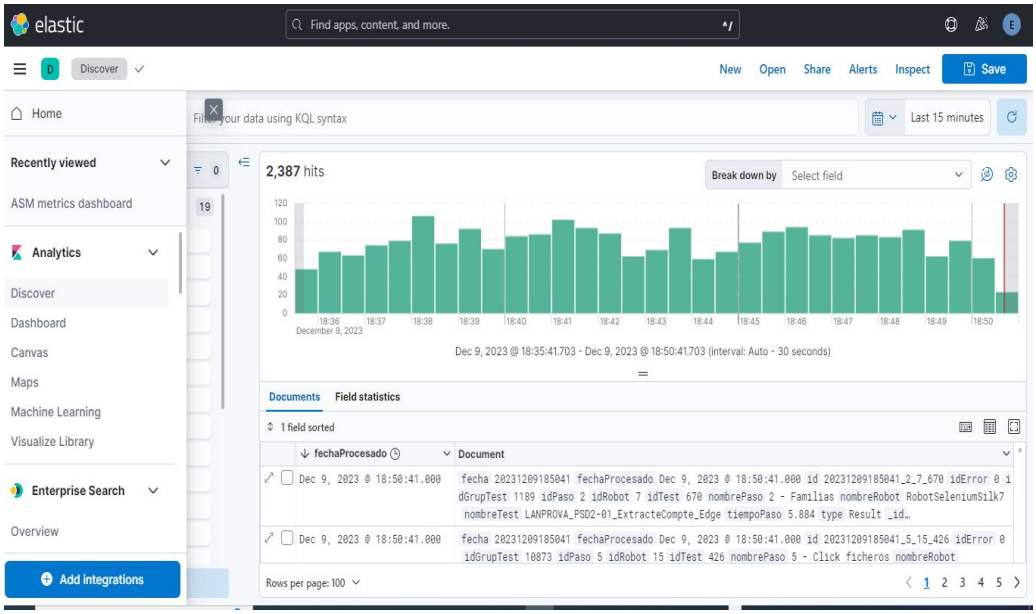

Figura 4.3: Kibana Discover.

```
# Modificar con el nombre del host específico si se
#quiere acceder por un nombre que no sea localhost
server.host: "node-1"

# Nombre que se le va a dar al servido
server.name: "node-1"

# Url's de los nodos de Elasticsearch
elasticsearch.hosts: ["https://node-1:9200", "https://node-2:9200", "https://node-3:9200"
    ]

# Modificar el usuario de acceso
elasticsearch.username: "kibana_system"
elasticsearch.password: "XXXX"
```

Una vez levantado el servicio 7.1.2, podremos acceder a la interfaz desde http:node-1:5601 y visualizar la información de los índices configurados. Desde Kibana también se pueden gestionar los usuarios, roles, índices o backups entre otras muchas funcionalidades.

Uno de los apartados de Kibana más empleados es el de **Analytics**, que permite analizar la información a tiempo real de los índices, crear visualizaciones o Dasboards, e incluso aplicar Machine Learning.

4.0.3. Apche Nifi

Apache NiFi es un sistema diseñado para gestionar flujos de datos, permitiendo la definición de procesos escalables de enrutamiento, transformación y gestión lógica mediante la construcción de grafos. Destaca por su interfaz de usuario web que facilita el diseño, control y monitoreo de los flujos de datos [3]. Una característica distintiva es la capacidad de programar flujos arrastrando y conectando componentes en la interfaz, eliminando la necesidad de conocimientos avanzados de programación. Aunque se clasifica como una herramienta ETL, no está optimizado para transformaciones complejas, sino más bien para automatizar ingestas de datos y realizar transformaciones y limpiezas sencillas, siendo comúnmente utilizado en entornos de Big Data.

Por ello, la elección de Apache NiFi como capa de preprocesamiento de datos se justifica por su enfoque centrado en la simplicidad y la facilidad de uso. Las personas encargadas de mantener esta capa pueden no tener conocimientos profundos en programación, y NiFi proporciona una interfaz gráfica intuitiva que permite diseñar y mantener flujos de datos de manera sencilla.

Los términos principales asociados a Apache Nifi son :

- **FlowFile**: Representa una sola pieza de datos en NiFi. Un FlowFile se compone de dos componentes: Atributos del FlowFile y Contenido del FlowFile. El contenido son los datos representados por el FlowFile. Los atributos son características que brindan información o contexto sobre los datos; se componen de pares clave-valor.

- **Procesador**: Es el componente de NiFi que se utiliza para escuchar los datos entrantes, extraer datos de fuentes externas, publicar datos en fuentes externas y enrutar, transformar o extraer información de FlowFiles. En este caso se han empleado “processors“ como PutElasticsearchJson, PublishKafka, ConsumeKafka, etc.

- **Relación**: Cada procesador tiene cero o más relaciones definidas. Estas relaciones se utilizan para indicar el resultado de procesar un FlowFile. Una vez que un procesador ha terminado de procesar un FlowFile, enrutará (o “transferirá”) el FlowFile a una de las relaciones. Se puede conectar cada una de estas Relaciones con otros componentes para especificar dónde debe ir el FlowFile a continuación según el resultado del procesamiento.

- **Conexión**: Une distintos componentes. Cada conexión consta de una o más relaciones. Para cada conexión que se dibuja, se puede determinar qué relaciones deben usarse. Esto permite que los datos se enruten de diferentes maneras según el resultado de su procesamiento. Cada conexión alberga una cola de FlowFiles.

Cuando un FlowFile se transfiere a una relación en particular, se agrega a la cola que pertenece a la conexión asociada.

En cuanto a la configuración (más información en 7.1.3), se han modificado los siguientes parámetros principales del fichero *nifi.properties*.

Debemos tener en cuenta que se ha levantado un Apache ZooKeeper 7.1.4 que Nifi usa como parte integral de su infraestructura para ofrecer capacidades de coordinación y gestión distribuida.

```
# Configurar en modo cluster
nifi.cluster.is.node=true
nifi.cluster.node.address=localhost
nifi.cluster.node.protocol.port=5888

# Añdir clave de seguridad común para todos los nodos
nifi.sensitive.props.key=XXX

# Indicar el string de conexión al zookeeper
nifi.zookeeper.connect.string=node-1:2181,node-2:2181,node-3:2181

# Indicar el hostname de la web y el puerto
nifi.web.https.host=localhost
nifi.web.https.port=443

# Configurar SSL
nifi.cluster.protocol.is.secure=true
```

En la figura 4.4, se muestran los dos grupos de procesos definidos para procesar por una parte los HAR, por otra parte las ejecuciones de tipo Result (es decir, una por cada paso del test) y las ejecuciones GroupedResult.

4.0.4. Apache Kafka

Apache Kafka [21] es una plataforma de gestión de colas, distribuida y de código abierto diseñada para desarrollar aplicaciones en tiempo real impulsadas por eventos. Se encarga de procesar flujos de datos, incluidos eventos, que pueden ir desde acciones de clientes, como realizar un pedido, hasta datos generados por máquinas, como ejecuciones de robots.

La plataforma permite la creación de aplicaciones que respondan y procesen estos flujos de datos en tiempo real con alta velocidad, fidelidad y precisión. Para ello Kafka proporciona tres capacidades clave: permitir que las aplicaciones **publiquen** o se **sus-**

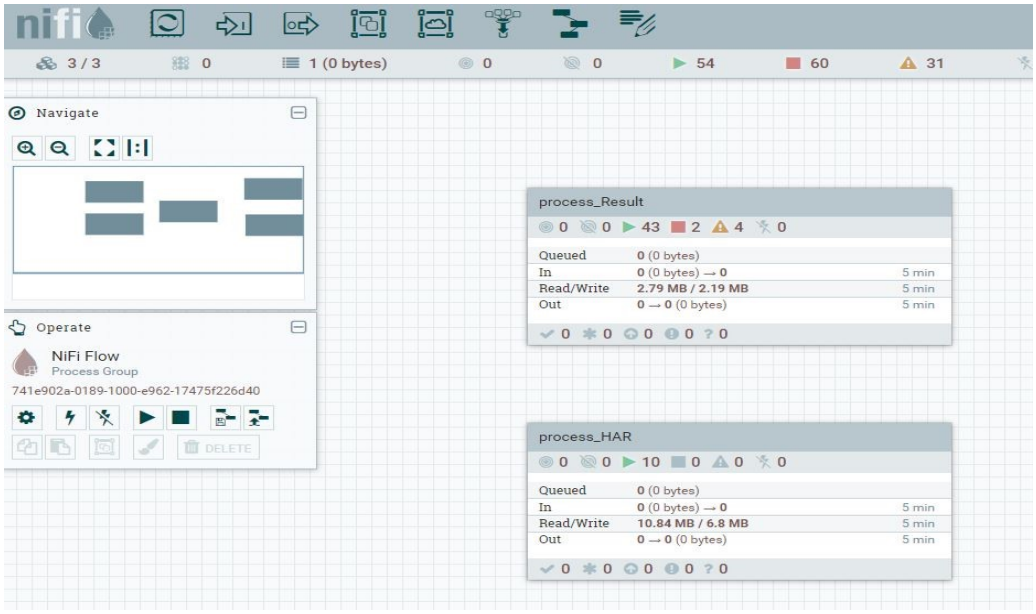

Figura 4.4: ProcessGroups definidos en Apache NiFi.

criban a flujos de datos, **almacenar** registros de manera precisa de manera tolerante a fallos y duradera, y **procesar** registros en tiempo real.

Los principales elementos (se reflejan en para interactuar con Kafka son:

- **Poducer**: Permite que las aplicaciones publiquen flujos de datos en Topics de Kafka, que han sido definidos con un nombre específico y que almacenan los datos en orden cronológico.
- **Consumer**: Permite que las aplicaciones se suscriban a Topics, ingieran y procesen el flujo en tiempo real o trabajen con registros pasados.
- **Streams**: Se basa en las API del Producer y del Consumer, proporcionando capacidades de procesamiento complejas para el procesamiento continuo de extremo a extremo.
- **Connect**: Permite que los desarrolladores construyan conectores, facilitando la integración de fuentes de datos en clústeres de Kafka.

Kafka se ha convertido en la plataforma de transmisión más utilizada, capaz de manejar billones de registros al día para organizaciones de diversas escalas, siendo un pilar para aplicaciones en tiempo real centradas en datos.

Al igual que Nifi, requiere de una instancia se Apache Zookeeper, por lo que hemos reutilizado el despliegue realizado para Nifi. Las modificaciones sobre el *server.properties* son:

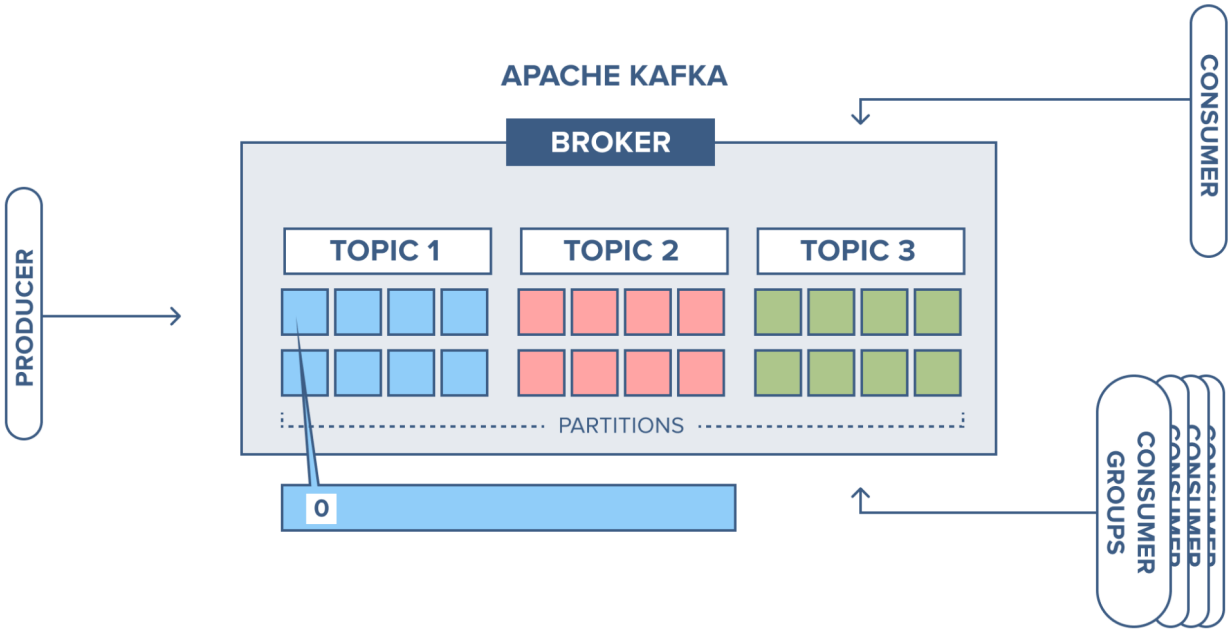

Figura 4.5: Funcionamiento de Apahe Kafka, tomado de [2] .

```
# Indicar un id al broker (uno distinto por cada nodo)
broker.id=0

# Indicar los puntos de conexión a los cuales el servidor Kafka estará escuchando
listeners=PLAINTEXT://node-1:9092

# Indicar ruta en la que se escribirán los datos
log.dirs=/data/kafka-logs

# Indicar string de conexión a Zookeeper
zookeeper.connect=node-1:2181,node-2:2181,node-3:2181
```

Además de los mencionado, hay dos parámetros bastante relevantes a la hora de configurar un cluster de Kafka, aunque en este caso no se han modificado los valores por defecto ya que no era necesario.

Esto parámetros son **log.retention.hours** y **log.retention.bytes**, que determinan cuánto tiempo y cuánto espacio en disco se deben retener los registros de logs en un clúster de Kafka. La configuración adecuada de estos parámetros permite equilibrar la necesidad de retener datos históricos con la capacidad de almacenamiento disponible en el sistema.

Estos parámetros se pueden definir tanto a nivel de cluster como a nivel de Topic, lo que aporta flexibilidad al sistema.

Finalmente, se han creado dos topic para almacenar los datos recibidos de los robots. Esto datos llegan en primer lugar a una API de Nodejs que dependiendo del endpoint al que se reciben, los distribuye en un Topic u otro.

- har-topic: Recoge información de los HAR.
- asm-raw-topic: Recoge la información a tiempo real de las ejecuciones de los test.

Capítulo 5

Desarrollo

En esta fase se describen los pasos seguidos para cumplir con los objetivos definidos en la descripción del proyecto. Sin embargo, antes de empezar, es necesario aclarar el flujo y contenido de los datos, desde el origen hasta su explotación final.

El sistema de **ASM** consiste en la recopilación de información proveniente de la ejecución de tests en seis entornos distintos:

- **Selenium (robots de web)**
- **MASM (robots de móviles)**
- **Selenium Grid (robots de web)**
- **SFTP (Secure File Transfer Protocol)**
- **API (Application Programming Interface)**
- **Robots SMS**
- **OASIS (recepción de mensajes PUSH o notificaciones)**

Concretamente, cada ejecución o tour consta de uno o más pasos (entre 3 y 10 pasos por ejecución aproximadamente) que se identifican con un robot y un grupo de ejecución (`idGrupTest`).

La arquitectura actual consta de tres componentes principales:

- **Máquina Virtual:** en la que se alojan los robots.
- **Servidor Web:** en el que se encuentra el web service que recoge la información de la ejecuciones y la herramienta de visualización actual, que se va a remplazar por esta.
- **Servidor de Base de Datos:** recopila el resultado de los tests. También aloja un sistema de alertas para los robots.

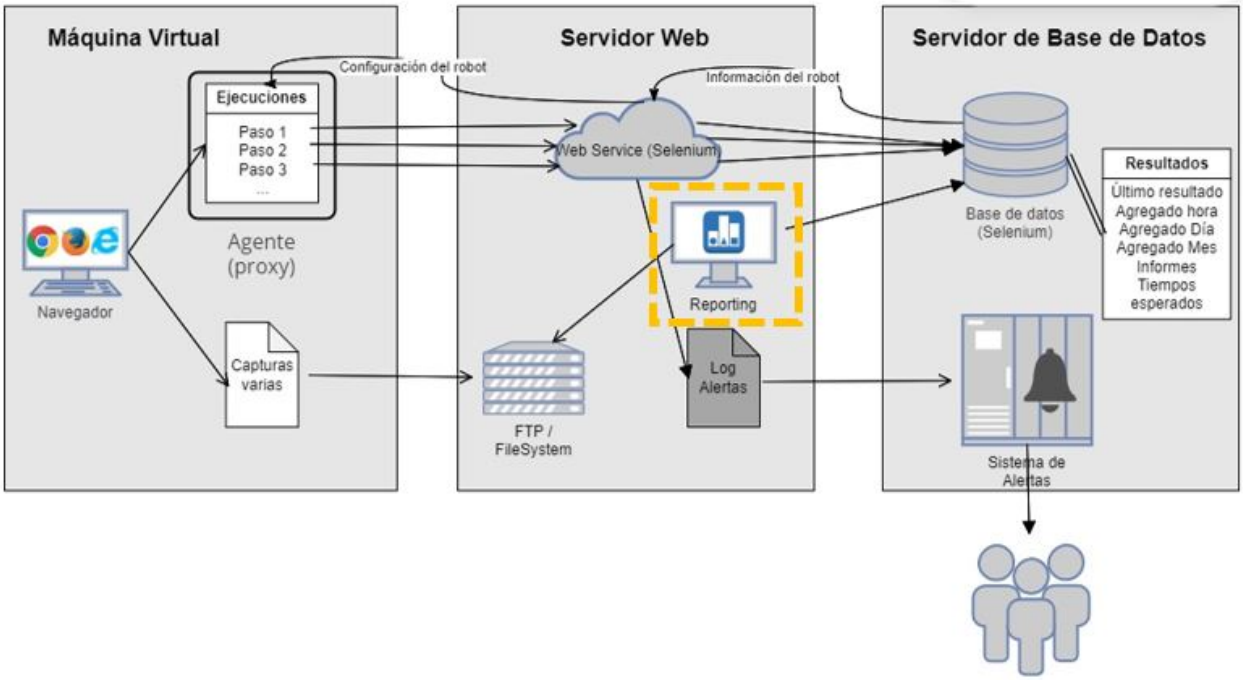

Figura 5.1: Funcionamiento del entorno Selenium.

Para cada entorno, el flujo de datos es similar, pero varía la información que se almacena en la base de datos. En la figura 5.1 se muestra el esquema del entorno Selenium, como ejemplo.

La información generada en estos entornos es enviada al servicio de NodeJS levantado para recepcionar estos mensajes y enviarlos al Topic de Kafka correspondiente. En concreto, los resultados de tipo HAR se envían al topic **har-topic** y los de tipo Result o GroupedResult se envían al topic asm-raw-topic, que hemos visto en la sección anterior 4.

Una vez almacenada la información en Nifi, es donde comienza el flujo de trabajo de nuestro entorno, tal y como se detalla en 5.2

5.0.1. Preprocesamiento con Apache Nifi

En esta primera fase se reciben cada uno de los registros generados por los web services y se limpian para su posterior procesamiento.

Las primeras fases de limpieza de datos son cruciales para garantizar la calidad, consistencia y confiabilidad de los datos. Permiten abordar errores, valores nulos, duplicados y sesgos, asegurando que los análisis subsiguientes se basen en datos precisos

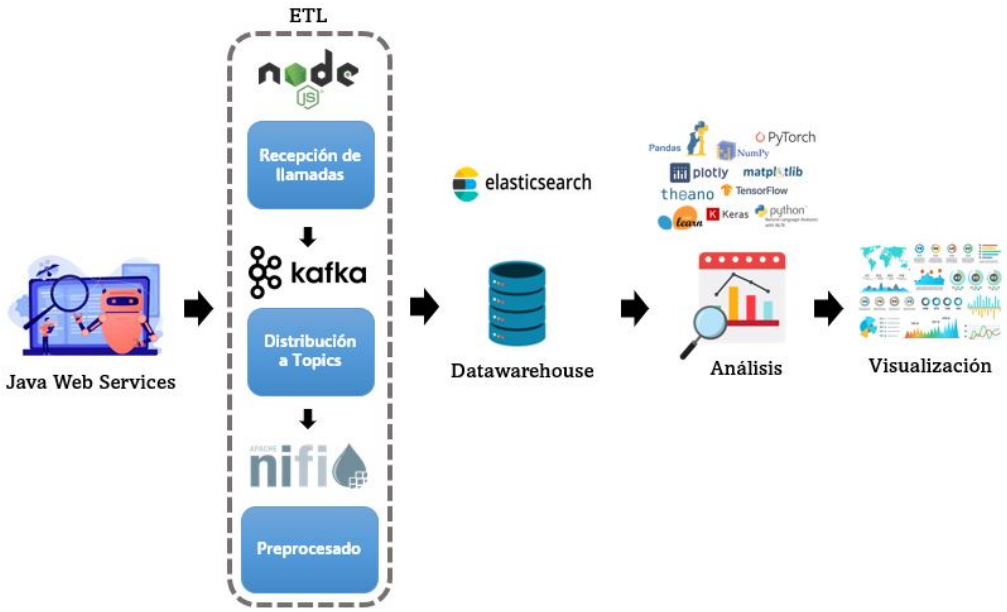

Figura 5.2: Flujo de los datos de monitorización.

y libres de inconsistencias. Esto es fundamental para obtener resultados confiables y tomar decisiones informadas en cualquier proyecto de análisis de datos.

Tal y como podemos ver en la imagen 5.3, primero se consumen los datos de la cola de kafka, luego se realizan una serie de reemplazos de caracteres especiales y finalmente se aplica un mapeo de campos para general el json con los campos deseados.

Sin embargo, no todos los registros tienen los mismo campos ni la misma información, sobre todo cuando se tratan de ejecuciones fallidas. Estas suelen traer además de los metadatos comunes sobre el test, el robot o el paso, información sobre el error, la ruta del error, la url o en caso de que sea en un entorno web, el elemento sobre el que se ha producido el error. Este último viene en un campo denominado ErrorSel (error de Selenium), que se emplea para filtrar los registros que traen este campo para posteriormente extraer el elemento que causa el error y así facilitar la detección de problemas.

Por ejemplo, un campo ErrorSel puede contener la siguiente información:

```
"[Test_429(Test_429): Unable to locate element: /html/body/div[3]/div[7]/div[5]/
   div[1]/div[2]/aFor documentation on this error, please visit: http://
   seleniumhq.org/exceptions/no_such_element.htmlBuild info: version: '3.6.0',
   revision: '6fbf3ec767', time: '2017-09-27T16:15:26.402Z'System info: host: '
   ISMSILKSEL40', ip: '10.210.5.40', os.name: 'Windows 10', os.arch: 'x86', os.
   version: '10.0', java.version: '1.8.0_371'Driver info: org.openqa.selenium.
   firefox.FirefoxDriverCapabilities [{moz:profile=C:\\Users\\ADMINI~1\\AppData\\
   Local\\Tempst_mozprofileXl3rRn, moz:geckodriverVersion=0.31.0, timeouts={
```

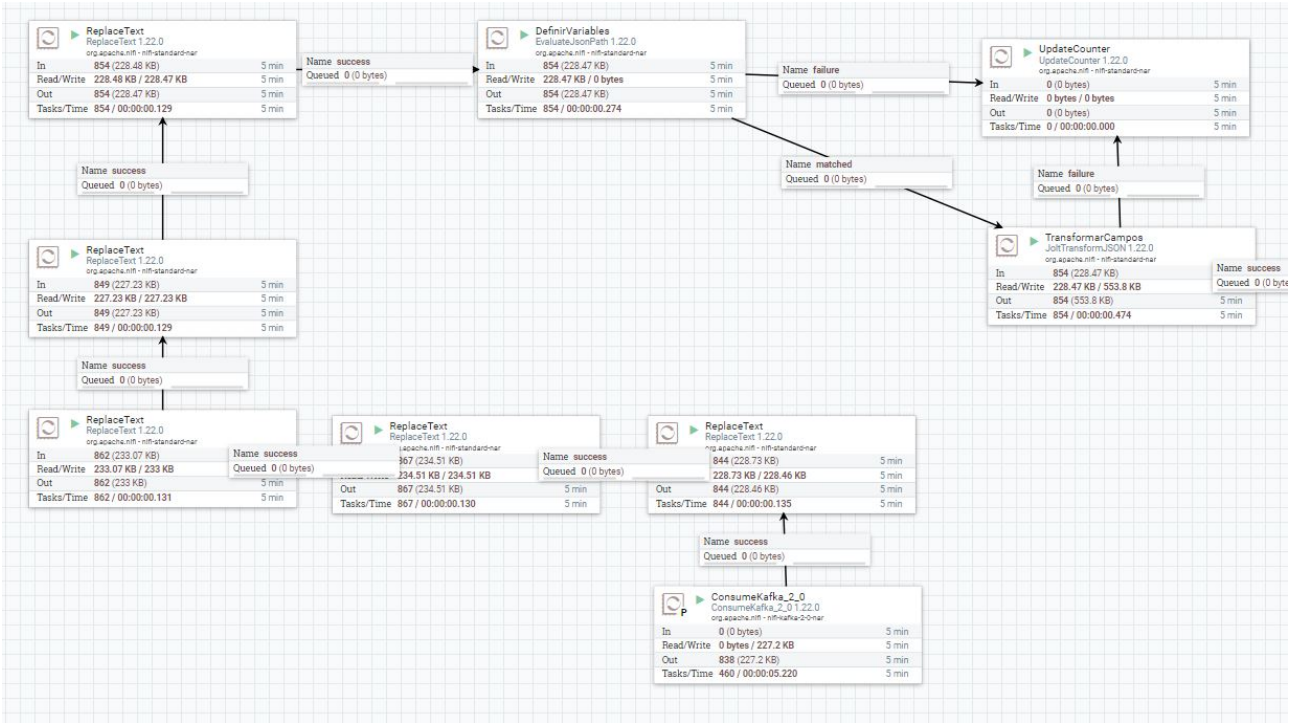

Figura 5.3: Primera etapa de carga y limpieza de datos desde Nifi.

```
implicit=0, pageLoad=300000, script=30000}, pageLoadStrategy=normal,
unhandledPromptBehavior=dismiss and notify, strictFileInteractability=false,
moz:headless=false, moz:windowless=false, platform=WINDOWS, proxy=Proxy(), moz
:accessibilityChecks=false, moz:useNonSpecCompliantPointerOrigin=false,
acceptInsecureCerts=true, browserVersion=113.0, moz:shutdownTimeout=60000, moz
:processID=6428, browserName=firefox, moz:buildID=20230504192738,
javascriptEnabled=true, moz:platformVersion=10.0, platformName=WINDOWS,
setWindowRect=true, moz:webdriverClick=true}]Session ID: 8ae5b495-cfd1-471d-93
01-1b44ce3c733d*** Element info: {Using=xpath, value=/html/body/div[3]/div[7]/
div[5]/div[1]/div[2]/a}]"
```

De toda esta información solo nos interesa */html/body/div[3]/div[7]/div[5]/div[1]/div[2]*. Para ello se han empleado una serie de processors que extraen el elemento y se escribe en un nuevo campo llamado elementError, tal y como se puede observar en 5.4.

Finalmente, todas las ejecuciones, ya sean correctas o incorrectas, se insertan en Elasticsearch. Se genera un identificador único combinando el ID del test, del paso y del robot. Es preferible generar este identificador personalizado, ya que, en caso de que un registro llegue dos veces, en lugar de crear dos entradas distintas, se actualiza el registro existente con los nuevos valores. Esto evita duplicados y mantiene la integridad

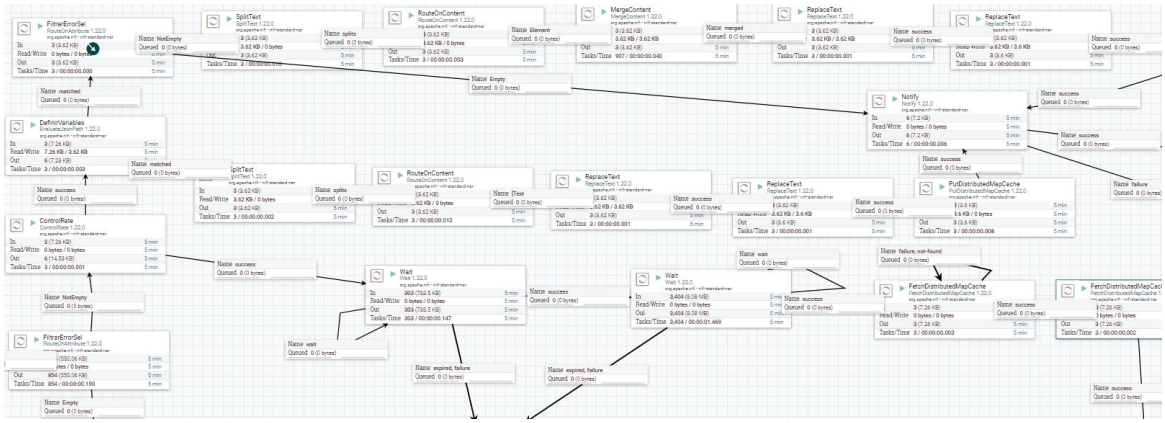

Figura 5.4: Extracción del elementError a partir del campo errorSel.

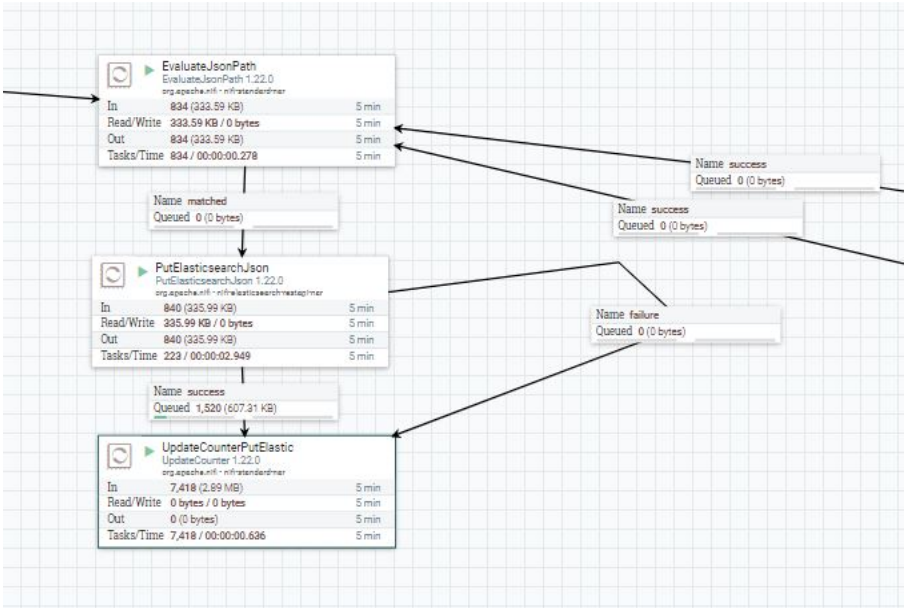

Figura 5.5: Escritura en Elasticsearch de los registros transformados.

de los datos en Elasticsearch.

5.0.2. Cálculo de agregados

Desde la web se presentará información de los tests agrupados por tramo horario, dependiendo del rango temporal que haya solicitado el usuario/a. Para ello, se realiza una query a elastic por test y por tramo horario:

```
query = {
        "query": {
            "bool": {
              "must": [
```

```
                {
                  "match": {
                    "idTest": str(idTest)
                  }
                },
                {
                  "range": {
                    "fechaProcesado": {
                      "gte": data_ini,
                      "lte": data_fi
                    }
                  }
                }
              ]
            }
        }
    }
```

El resultado lo convertimos en un pandas DataFrame y filtramos por los registros cuyo *tipo="GroupedResult",* pues si no lo hiciéramos estaríamos calculando múltiples resultados (uno por paso) para una misma ejecución.

Con este DataFrame calculamos el porcentaje de acierto del test en el periodo dado, el porcentaje de error y el número total de ejecuciones. Además nos quedamos con el resultado de la última ejecución y la duración de la misma.

```
aggregation_dict = get_aggregated_data_for_test(test_results, current_date,
    previous_date)
aggregation_dict['fechaUltimaEjecucion'] = str(last_execution_date).replace(" ",
    "T")
aggregation_dict['duracionUltimaEjecucion'] = str(last_execution_duration)
aggregation_dict['tags'] = tags
aggregation_dict['entorno'] = enviroment
aggregation_dict['nombreTest'] = test_name
aggregation_dict['nombreRobot'] = robot_name
aggregation_dict['errorUltimaEjecucion'] = failed_execution_state
```

Esta información se pinta en a web tal y como se ve en la siguiente imagen 5.6 .

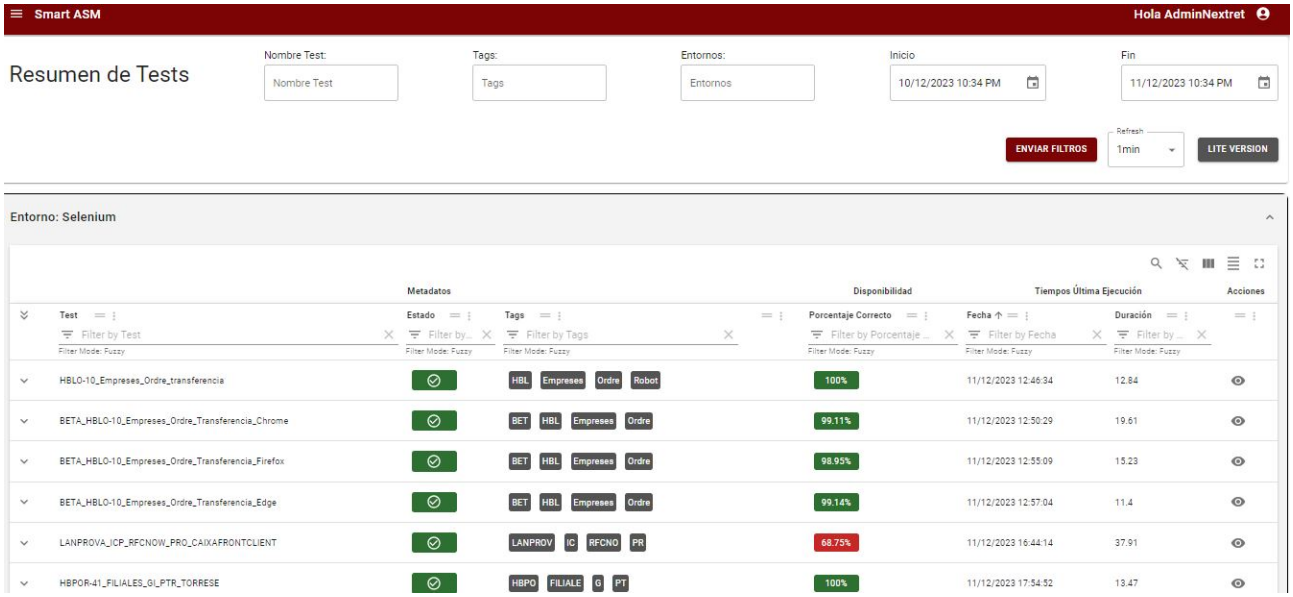

Figura 5.6: Visualización de ejecuciones agrupadas.

5.0.3. Visualización de datos

Como se ha mencionado anteriormente, la efectiva presentación de la información desempeña un papel crucial en la comprensión y análisis de datos. En esta sección, nos centraremos en la 'Visualización de datos', una herramienta poderosa que permite representar de manera gráfica y comprensible los patrones y tendencias inherentes a los conjuntos de datos. La visualización no solo simplifica la interpretación de la información, sino que también facilita la identificación de insights clave, que en este caso pueden audar a detectar errores en los sistemas monitorizados.

A continuación, se presentan algunas de las gráficas realizadas para visualizar la información de los tests de manera efectiva, contribuyendo así a una comprensión más profunda y rápida de los resultados.

- **KPIs sobre la ejecución del test**

  En este caso es fundamental identificar y analizar indicadores clave que proporcionen una visión integral de la calidad y eficiencia del proceso. Aquí se presentan algunos KPIs relevantes para la ejecución de los tests:

  1. **Tiempo Medio de Duración del Test:**
     - **Definición:** El tiempo promedio que toma la ejecución de un test.
     - **Importancia:** Indica la eficiencia de las ejecuciones y puede ayudar a identificar posibles cuellos de botella.

2. **Variación del Tiempo de Duración:**
    - **Definición:** La variación en el tiempo de duración de los tests respecto al mismo periodo de tiempo anterior.
    - **Importancia:** Permite identificar cambios significativos en la eficiencia de ejecución y anticipar posibles problemas.
3. **Disponibilidad Media:**
    - **Definición:** El porcentaje de ejecuciones correctas respecto al total de ejecuciones.
    - **Importancia:** En caso de valores muy bajos, indica posibles problemas en los casos en los que se ejecutan los tests.
4. **Número de Errores:**
    - **Definición:** La cantidad total ejecuciones erróneas encontradas.
    - **Importancia:** Proporciona una medida directa de la estabilidad y calidad del software.
5. **Variación en la Disponibilidad por Día o por Horas:**
    - **Definición:** La variación en el porcentaje de ejecuciones correctas durante el período filtrado.
    - **Importancia:** Nos permite identificar tramos horarios clave para la detección de problemas.

Estos KPIs proporcionan una base sólida para evaluar la ejecución de tests desde diversas perspectivas. La gestión continua de estos indicadores permite la identificación proactiva de áreas de mejora sobre los entornos monitorizados. La respresentación visual se puede ver en 5.7

- **KPIs sobre los pasos de cada ejecución**

  Teniendo en cuenta los pasos que componen una ejecución, es muy interesante analizarlo con detalles porque cada uno testea un componente distinto de una aplicación, por ejemplo, por lo que nos puede dar información sobre qué apartado de una web está fallando. Normalmente los aparatados que más veces fallan son el Login y el Logout pero puede haber otro componente que por algún motivo, que puede ser una actualización de código, este fallando constantemente :

  1. **Duración Media por Paso en el Periodo Temporal:**

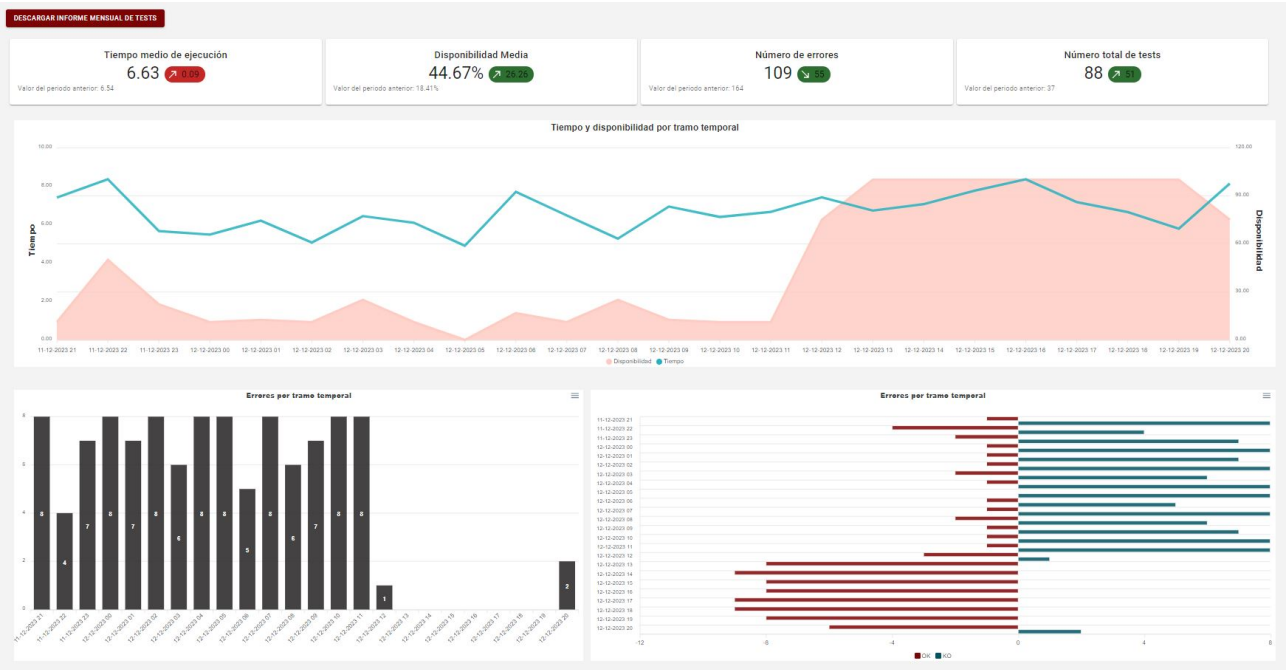

Figura 5.7: KPIs sobre la ejecución del test.

- **Definición:** El tiempo promedio que toma la ejecución de cada paso de prueba durante un periodo temporal específico.
- **Importancia:** Indica la eficiencia de la ejecución de pasos individuales y permite identificar posibles áreas de mejora.

2. **Porcentaje de Ejecución Correcta por Paso y Día:**
   - **Definición:** El porcentaje de ejecuciones correctas respecto al total de ejecuciones para cada paso, desglosado por día.
   - **Importancia:** Evalúa la fiabilidad y éxito de cada paso en diferentes momentos, permitiendo detectar patrones de desempeño diario.
3. **Número de Errores por Paso:**
   - **Definición:** La cantidad total de errores encontrados durante la ejecución de cada paso de prueba.
   - **Importancia:** Proporciona una métrica específica sobre la estabilidad y calidad de cada paso.

1. **KPIs avanzados**

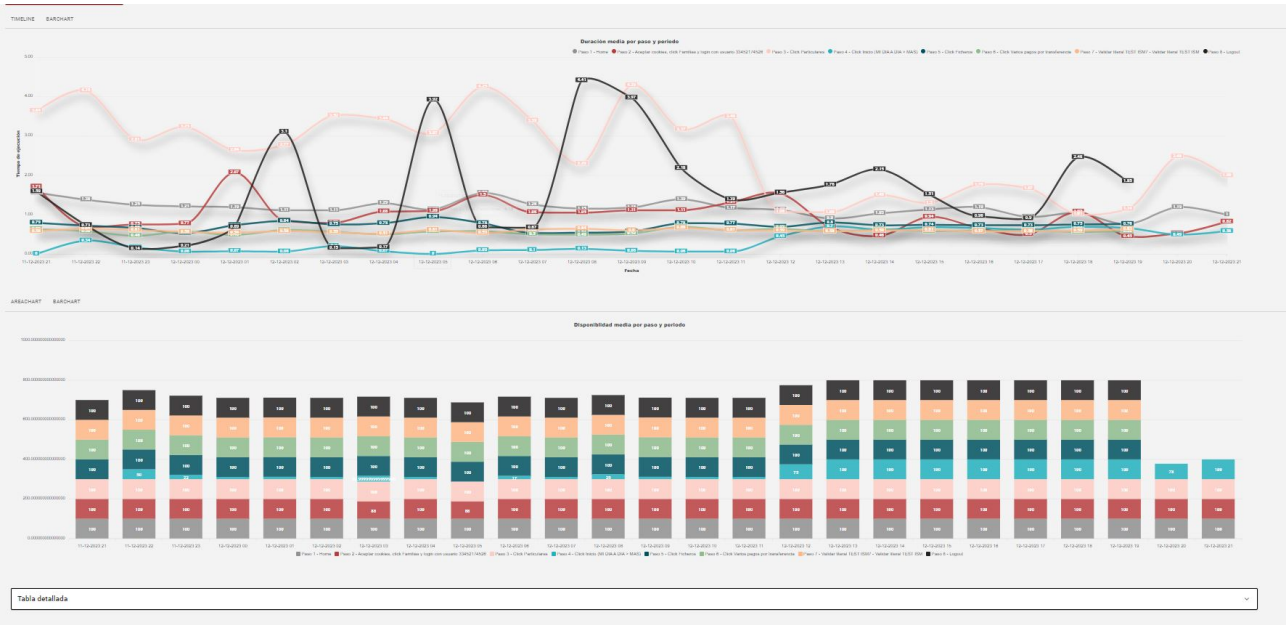

Figura 5.8: KPIs sobre los pasos de cada ejecución.

Por último, se han aplicado KPIs avanzados que utilizan técnicas de análisis de texto y web scraping para extraer información oculta de las trazas HAR y de las URL fallidas. Estos KPIs avanzados incluyen:

a) **Análisis de trazas HAR:**

- **Extracción de Información Oculta:** Aplicación de técnicas de análisis de texto en las trazas HAR para identificar patrones, problemas o características específicas.
- **KPIs Resultantes:** Nuevos KPIs derivados del análisis de las trazas HAR, como la detección de comportamientos inesperados o patrones de rendimiento.

b) **Análisis de URL Fallidas:**

- **Web Scraping:** Utilización de técnicas de web scraping para recopilar información adicional sobre las URL en las que ha fallado alguna ejecución del test en el tramo horario dado.
- **KPIs Resultantes:** Datos mejorados sobre las características y contenidos de las páginas que generaron errores, proporcionando una visión más detallada del contexto del fallo.

c) **Clasificación de HTML mediante Deep Learning:**

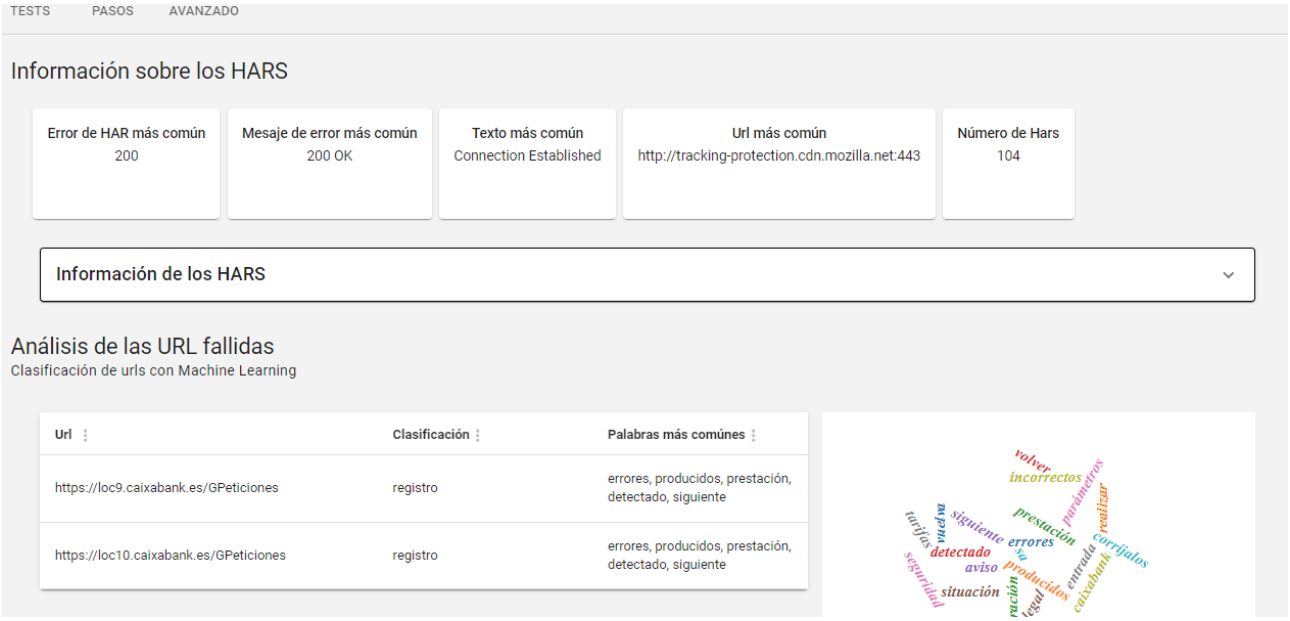

Figura 5.9: Visualizaciones resultante del análisis de textos.

- **Uso de Algoritmos de Deep Learning:** Implementación de algoritmos de Deep Learning para clasificar el tipo de página que se estaba consultando cuando se produjo un error.
- **KPIs Resultantes:** Categorización de los errores en función del HTML de la página, lo que facilita la identificación de patrones específicos relacionados con categorías de contenido.

Estos KPIs avanzados, tal y como se pueden observar en , brindan una capa adicional de análisis, revelando información oculta en las trazas HAR y en las URL fallidas. La combinación de técnicas de análisis de texto y web scraping, junto con el poder de clasificación de los algoritmos de Deep Learning, enriquece significativamente la comprensión de los errores y la detección temprana de problemas.

5.0.4. Aplicación de Procesamiento del Lenguaje Natura sobre el texto libre en el HTML de las url.

En el panorama de la Ciencia de Datos, una de las tareas fundamentales es la clasificación de texto, que es ampliamente utilizada en el procesamiento del lenguaje natural (NLP) con aplicaciones que abarcan diversos ámbitos. La eficiencia de un clasificador de texto desempeña un papel crucial al categorizar automáticamente y de forma significativa mediante el uso de algoritmos de NLP.

Este tipo de problemas se tratan de un ejemplo de tarea supervisada de aprendizaje automático, utilizando conjuntos de datos etiquetados que contienen documentos de texto y sus respectivas etiquetas para entrenar un clasificador. El objetivo es capacitar al algoritmo para generalizar y categorizar con precisión nuevos documentos de texto no vistos anteriormente. Comúnmente se emplean técnicas como Clasificador Naive Bayes, Clasificador Lineal, Máquinas de Soporte Vectorial (SVM) o Redes Neuronales Profundas (DNN).

Mientras que los principios de la clasificación de texto proporcionan una base sólida, la integración del Procesamiento del Lenguaje Natural con datos HTML introduce un conjunto único de desafíos y oportunidades. Simultáneamente, la técnica de Web Scraping se vuelve fundamental para extraer información valiosa de los sitios web.

El web scraping, o extracción de datos web, implica la extracción automatizada de datos de sitios web utilizando software que simula la navegación web humana. Las técnicas para web scraping incluyen copiar y pegar humano, coincidencia de patrones de texto, programación HTTP, y análisis de HTML y DOM, agregación vertical, reconocimiento de anotaciones semánticas y análisis de páginas web mediante visión por computadora.

Es por ello por lo que en este proyecto se ha optado por un enfoque integral, dividido en tres módulos distintos:

1. **Extracción de Datos (Data Scraping):** Se emplean técnicas para recopilar eficientemente datos de los sitios web.

```
url_classficaion_list = []
processed_url = {}
# Recorrer los test
for hit in unique_idTest_list["aggregations"]["unique_idTest"]['buckets']:
    idTest = hit['key']
    # Recorrer los test
    data_current_df = get_raw_asm_test_data(es, index_name, idTest,
        datetime_ini_str, datetime_fi_str)

    for index, row in data_current_df.iterrows():
        # Filtrar las filas que tengan valor para el campo urlFailed
        if row['urlFailed'] != "":
            main_url = row['urlFailed']
            base_url =  main_url.split('?')[0]
            # Si todavía no se ha procesado la url, se procesa
            if not base_url in processed_url.keys():
                t0 = time.time()
```

```
                try:
                    # Recogemos el html, guardando el tiempo de respuesta
                        para detenr el programa temporalmente en función del
                        mismo
                    # De esta forma se evita saturar la web consultada
                    url_response = requests.get(main_url)
                    url_response = requests.get(main_url, verify=False,
                        cookies=url_response.cookies)
                    response_delay = time.time() - t0
                except:
                    print("error")
                    break
            # Obtener el contenido
            soup = BeautifulSoup(url_response.content)
            main_div = soup.find_all(class_="main")
            if(len(main_div) > 0):
                main_div = main_div[0]
            else:
                main_div = soup
            texts = main_div.findAll(text=True)
            text_from_html = ' '.join(texts)
            # Clasificar según las categorías
            category = get_text_category(text_from_html)
            url_classficaion_dict = {
                "tokenized_source": text_from_html[0:800],
                "category": category,
                "url": main_url
            }
            processed_url[base_url] = category
            url_classficaion_list.append(url_classficaion_dict)

            time.sleep(1 * response_delay)
```

2. **Clasificación basada en Palabras Clave:** Permite crear un conjunto de datos de entrenamiento clasificando los datos extraídos mediante palabras clave. Luego el resultado de esta clasificación, se revisa y se corrige en los casos que sea necesario.

```
# Página de Inicio
home_page_keywords = ["banca", "home", "promociones", "productos", "
```

```
    financieros", "bancarios", "servicios", "noticias", "banco", "ahorro", "
    préstamos"]

# Formularios de Inicio de Sesión
login_keywords = ["sesión", "login", "usuario", "contraseña", "password", "
    acceso", "credenciales", "identificador", "entrar"]

# Formularios de cierre de Sesión
logout_keywords = ["sesión", "logout", "cierre", "salir"]

# Sección de Registro
register_keywords = ["registro", "crear", "personal", "datos", "abrir"]
# Información de Productos y Servicios
services_keywords = ["ahorro", "cuenta", "corrientes", "préstamos", "
    tarjetas", "casa", "nómina", "fondos", "bolsa", "mercados", "
    sostenibilidad", "depósitos",
                     "crédito", "inversiones", "hipotecas", "donaciones", "
                        jubilación", "inversión", "renting", "seguros", "
                        salud", "vida", "hogar", "coche", "moto"]

all_keywords_processor=KeywordProcessor()
for word in keywords:
    all_keywords_processor.add_keyword(word)

home_page_keywords_processor = KeywordProcessor()
for word in home_page_keywords:
    home_page_keywords_processor.add_keyword(word)

login_keywords_processor =KeywordProcessor()
for word in login_keywords:
    login_keywords_processor.add_keyword(word)

logout_keywords_processor =KeywordProcessor()
for word in logout_keywords:
    logout_keywords_processor.add_keyword(word)

register_keywords_processor=KeywordProcessor()
for word in register_keywords:
    register_keywords_processor.add_keyword(word)
```

```
services_keywords_processor=KeywordProcessor()
for word in services_keywords:
    services_keywords_processor.add_keyword(word)
```

3. **Aplicación de Redes Neuronales y Otros Modelos para la Clasificación de Texto:** Se han implementando diversos modelos avanzados, incluidas redes neuronales, para realizar la clasificación de texto basada en el conjunto de datos enriquecido.

   Una vez implementados, la comparación de los resultados obtenidos de diversos modelos se convierte en una fase crucial. Este proceso de comparación tiene como objetivo identificar el modelo que mejor se ajusta a las necesidades específicas del problema de clasificación de texto. Las métricas de evaluación, como la precisión, el recall y la F1-score, son esenciales para medir el rendimiento de cada modelo.

   Además, para mejorar la eficiencia y confiabilidad de nuestros modelos de clasificación de texto, hemos aplicado técnicas avanzadas durante el proceso. En particular, se ha llevado a cabo un proceso de hiperparametrización exhaustiva, que consiste en ajustar los parámetros del modelo para optimizar su rendimiento específicamente en nuestro conjunto de datos.

   Adicionalmente, hemos implementado la validación cruzada en un esquema de 5 pliegues. Esta estrategia divide nuestro conjunto de datos en cinco partes diferentes, realizando múltiples iteraciones de entrenamiento y evaluación. Esto no solo permite una evaluación más robusta del rendimiento del modelo, sino que también ayuda a mitigar el riesgo de sobreajuste y garantiza que el modelo seleccionado sea generalizable a datos no vistos.

   La combinación de la hiperparametrización y la validación cruzada ha sido fundamental para obtener un modelo de clasificación de texto que no solo se destaque en el conjunto de datos de entrenamiento, sino que también tenga la capacidad de generalizar efectivamente en situaciones del mundo real. Estas técnicas avanzadas fortalecen la confiabilidad y la eficacia de nuestro enfoque, proporcionando resultados más sólidos y adaptados a las complejidades de nuestro problema específico de clasificación de texto.

   En concreto se han generado los siguientes modelos:

   - **Redes Neuronales Profundas (Word embedding + LSTM)**. Anexo 7.2.1

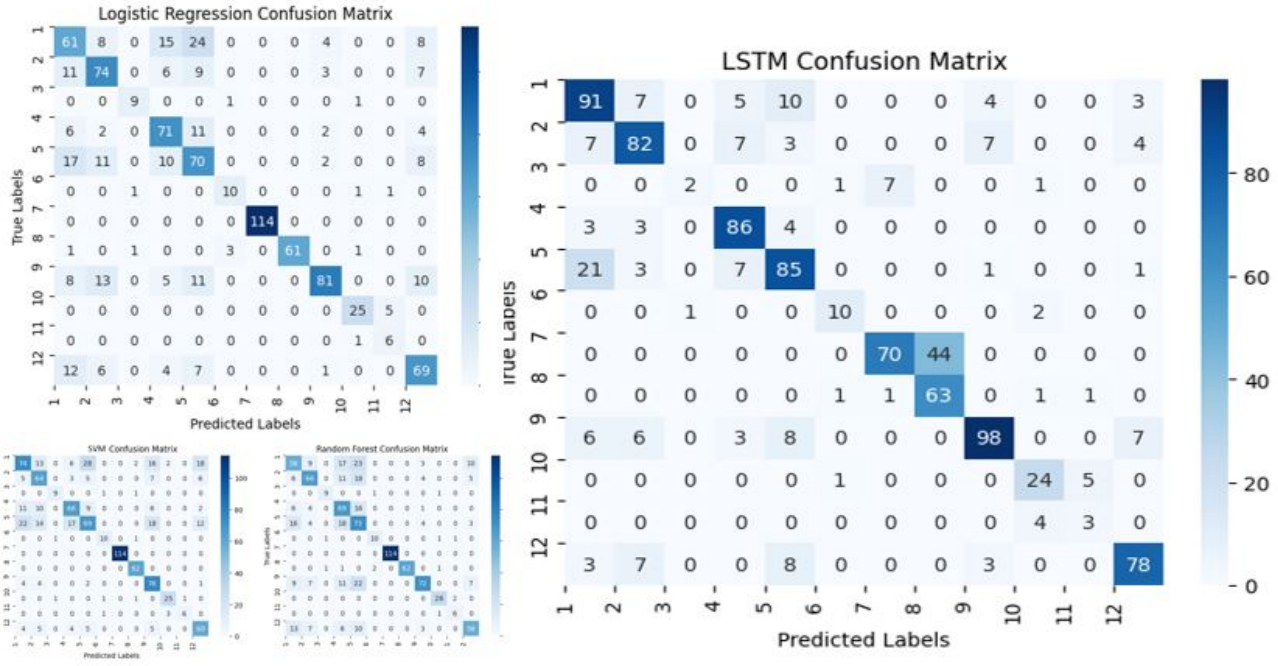

Figura 5.10: Matriz de confusión de los modelos generados.

- **Regresión Lineal**. Anexo 7.2.2
- **Random Forest**. Anexo 7.2.3
- **Máquina de Soporte Vectorial (SVM)**. Anexo 7.2.4

Los resultados se pueden comparar en la figura 5.10.

Modelo	Accuracy
Word embedding + LSTM	0.76
Regresión Lineal	0.71
Random Forest	0.69
SVM	0.68

Cuadro 5.1: Resultado de los modelos generados.

A través de la fusión de estos módulos, nuestro objetivo es mostrar la sinergia entre la extracción de datos web, la clasificación basada en palabras clave y la aplicación de modelos de vanguardia para la clasificación de texto en el contexto del contenido HTML. Este enfoque holístico está diseñado para descubrir información oculta dentro de los datos web y mejorar las capacidades de clasificación de texto en un entorno centrado en la web. Finalmente, esta clasificación se ha representado visualmente, tal y como se ve en la figura 5.9.

Capítulo 6

Conclusiones y Líneas Futuras

Tal y como se comentaba en la sección de Objetivos, en este **Trabajo de Fin de Máster** se buscaba desarrollar una herramienta integral de monitorización avanzada que permitiera la detección temprana de problemas, la identificación precisa del origen de errores, y una mejora en la eficiencia en el análisis de problemas.

Teniendo en cuenta estos propósitos, podríamos afirmar que el resultado ha sido exitoso, pues se ha logrado cumplir con los objetivos establecidos. Esto ha sido posible gracias a la aplicación de **técnicas avanzadas** de procesado de datos e Inteligencia Artificial, que han permitido abordar diversos aspectos como el análisis de textos, la clasificación o la extracción de patrones “ocultos“ en los datos. Además, el uso de herramientas modernas de **visualización** nos ha proporcionado la capacidad de extraer conclusiones significativas de los datos analizados, contribuyendo así a una toma de decisiones más informada y precisa.

Por otro lado, es innegable que este proyecto ha sido sumamente desafiante y motivador al intentar abordar diversas tareas en un período de tiempo limitado. Considero que aún existen áreas con un potencial significativo para profundizar y mejorar. Especialmente, se abre la posibilidad de enriquecer el proyecto integrando más técnicas de inteligencia artificial, como la detección de patrones a partir de imágenes proporcionadas por los robots, o explorar la detección de posibles fallos en los sistemas mediante un análisis más profundo de series temporales. Este enfoque ampliado podría no solo perfeccionar la eficacia del sistema actual, sino también abrir nuevas oportunidades para la innovación y la mejora continua.

Para concluir, en cuanto a los objetivos del **máster en Ciencia de Datos**, este proyecto se destaca por ser integral y abarcar una amplia gama de conocimientos adquiridos a lo largo del máster. Al aplicar conceptos fundamentales de la **ciencia de datos**, abordar la **tipología y ciclo de vida de los datos**, diseñar arquitecturas de bases de **datos no relacionales** y emplear técnicas de **aprendizaje automático**, demuestra una comprensión profunda y aplicada de los fundamentos del campo.

La inclusión de la **visualización de datos** y la creación de un entorno de **análisis de big data** subraya la versatilidad del proyecto, ya que integra múltiples disciplinas del máster. Este enfoque completo no solo evidencia la aplicación práctica de los co-

nocimientos teóricos adquiridos, sino que también resalta la capacidad para abordar **desafíos multidisciplinarios** y crear soluciones que aprovechan diversas áreas de estudio. En resumen, este trabajo no solo cumple con los objetivos específicos del proyecto, sino que también demuestra la amplitud y profundidad de los conocimientos obtenidos durante el máster en diversas áreas relevantes.

Bibliografía

[1] 8 steps in the data life cycle | hbs online.

[2] apache-kafka-partition.png (1999×1043).

[3] Apache nifi: Introducción 2023 - aprender big data.

[4] Cap theorem and distributed database management systems | by syed sadat nazrul | towards data science.

[5] Ciudades - desarrollo sostenible.

[6] Classified website scraping.

[7] Consideraciones para big data: arquitectura y enfoque.

[8] El elastic stack: Elasticsearch, kibana, beats y logstash | elastic.

[9] elasticsearch-image-1-1024x643.png (1024×643).

[10] Elasticsearch: What it is, how it works, and what it's used for - knowi.

[11] Getting started with heartbeat. monitor uptime for your applications. . . | by vikas yadav | devops dudes | medium.

[12] Industrial classification of websites by machine learning with hands-on python | by ridham dave | towards data science.

[13] Infraestructura - desarrollo sostenible.

[14] Infraestructura - desarrollo sostenible.

[15] La importancia de los ods para las empresas| deloitte españa.

[16] Objetivos de desarrollo sostenible (ods) | cepal.

[17] Salud - desarrollo sostenible.

[18] The 17 goals | sustainable development.

[19] Understanding the layout of webpages using automatic zone recognition | by julien dumazert | contentsquare engineering | medium.

[20] Web page content analysis made easy with streamlit: A step-by-step guide | by marcello dichiera | geek culture | medium.

[21] What is apache kafka? | ibm.

[22] What is kibana used for? –10 important features to know.

[23] Data management in cloud environments: Nosql and newsql data stores. 2013.

[24] Technical report, 2023.

[25] Amazon ec2. Technical report, 2023.

[26] Amazon elastic compute cloud. Technical report, 2023.

[27] Amazon elastic compute cloud architecture. Technical report, 2023.

[28] Apache hadoop. Technical report, 2023.

[29] Apache hadoop ecosystem. Technical report, 2023.

[30] Aws ecs. Technical report, 2023.

[31] Aws ecs. Technical report, 2023.

[32] Aws lambdal. Technical report, 2023.

[33] Benefitting from big data leveraging unstructured data capabilities for competitive advantage. Technical report, 2023.

[34] Challenges of big data: Basic concepts, case study, and more. 2023.

[35] Grafana. Technical report, 2023.

[36] Splunk. Technical report, 2023.

[37] Análisis en aws. Technical report, 2023.

[38] Marcin Bajer. Building an iot data hub with elasticsearch, logstash and kibana. In *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 63–68, 2017.

[39] Payal M. Bante and K. Rajeswari. Big data analytics using hadoop map reduce framework and data migration process. In *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, pages 1–5, 2017.

[40] Eugen Betke and Julian Kunkel. Real-time i/o-monitoring of hpc applications with siox, elasticsearch, grafana and fuse. pages 174–186, 10 2017.

[41] Xiang Deng, Prashant Shiralkar, Colin Lockard, and Binxuan Huang. Dom-lm: Learning generalizable representations for html documents; dom-lm: Learning generalizable representations for html documents.

[42] Ferran Garrido Fernández. Arquitectura big data de ingesta en real time. Technical report, 2017.

[43] Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, Aleksandra Faust, and Google Research. Understanding html with large language models.

[44] Robin Hecht and Stefan Jablonski. Nosql evaluation: A use case oriented survey. In *2011 International Conference on Cloud and Service Computing*, pages 336–341, 2011.

[45] Han Liu Jianqing Fan, Fang Han. Challenges of big data analysis. *National Science Review*, 2014.

[46] Parag Mulendra Joshi and Sam Liu. Web document text and images extraction using dom analysis and natural language processing. *DocEng'09 - Proceedings of the 2009 ACM Symposium on Document Engineering*, pages 218–221, 2009.

[47] Anita Kamdi. Big data quality assurance and testing framework. Technical report, 2018.

[48] Garrett McGrath and Paul R. Brenner. Serverless computing: Design, implementation, and performance. pages 405–410, 2017.

[49] Alex Owen-Hill. 5 issues to solve before monitoring robot performance. 2020.

[50] Fiorella Piriz Sapio. Deep learning and its applications in multiple sequence alignment. Technical report, 2021.

[51] Alexandru Adrian TOLE. Big data challenges. Technical report, 2013.

[52] Òscar Romero and Marta Oliva. Distributed databases.

Capítulo 7

Anexos

7.1. Guías de instalación

7.1.1. Elasticsearch

```
# INSTALACION ELASTIC 8.9
vi /etc/yum.repos.d/elastic.repo
yum install elasticsearch
tail -f /var/log/elasticsearch/cluster.log

systemctl stop elasticsearch

uname -a
update-crypto-policies --set LEGACY

systemctl status firewalld
vi /etc/elasticsearch/jvm.options
vi /etc/elasticsearch/elasticsearch.yml
systemctl enable elasticsearch
systemctl start elasticsearch

# CONFIGURACIÓN
vi /etc/elasticsearch/elasticsearch.yml

  # Establecer un nombre al cluster (debe ser el mismo en todos los nodos)
  cluster.name: cluster-name

  # Establcer un nombre al nodo (debe ser descriptivo)
  node.name: node-1

  # Modificar con el nombre del host específico si se
  #quiere acceder por un nombre que no sea localhost
  network.host: node-1

  # Modificar con el nombre de todos los host del cluster
  discovery.seed_hosts: ["node-1", "node-2", "node-3"]

```

```
33    # Añadir la ruta en la que se quieren guardar los datos
34    path.data: /data/elasticsearch
35
36    # Habilitar comunicación por SSL
37    xpack.security.transport.ssl:
38      enabled: true
39      verification_mode: certificate
40      client_authentication: required
41      keystore.path: elastic-certificates.p12
42      truststore.path: elastic-certificates.p12
43
44
45  tail -f /var/log/elasticsearch/cluster.log
46
47  scp /etc/elasticsearch/elasticsearch.yml node-2:/etc/elasticsearch/elasticsearch.yml
48  scp /etc/elasticsearch/elasticsearch.yml node-3:/etc/elasticsearch/elasticsearch.yml
49
50  ## ERROR
51  failed to load SSL configuration [xpack.security.transport.ssl] - cannot read configured
      [PKCS12] keystore (as a truststore) [/etc/elasticsearch/certs/transport.p12] - this
      is usually caused by an incorrect password; (no password was provided)
52
53  https://www.elastic.co/guide/en/elasticsearch/reference/8.8/security-basic-setup.html#
      encrypt-internode-communication
54  https://www.elastic.co/guide/en/elasticsearch/reference/current/security-basic-setup-
      https.html#encrypt-kibana-elasticsearch
55  cd /usr/share/elasticsearch/
56
57
58  # GENERAR CERTIFICADOS
59
60  # Hacer esto solo en el un nodo
61
62    /usr/share/elasticsearch/bin/elasticsearch-certutil ca
63
64    /usr/share/elasticsearch/bin/elasticsearch-certutil cert --ca elastic-stack-ca.p12
65
66    /usr/share/elasticsearch/bin/elasticsearch-certutil http
67
68  # Repetir esto en todo los nodos
69
70    /usr/share/elasticsearch/bin/elasticsearch-keystore add xpack.security.transport.ssl.
        keystore.secure_password
71
```

```
  /usr/share/elasticsearch/bin/elasticsearch-keystore add xpack.security.transport.ssl.truststore.secure_password

  /usr/share/elasticsearch/bin/elasticsearch-keystore add xpack.security.http.ssl.keystore.secure_password

  systemctl retart elasticsearch

# GENERAR USUARIO Y PASSWORD DE ACCESO
/usr/share/elasticsearch/bin/elasticsearch-setup-passwords auto
/usr/share/elasticsearch/bin/elasticsearch-reset-password -u elastic-user

# ACCESO A LA API
https://node-1:9200
```

7.1.2. Kibana

```
# INSTALACION KIBANA 8.9
yum install kibana
# Crear usuario kibana_system
/usr/share/elasticsearch/bin/elasticsearch-reset-password -u kibana_system

# CONFIGURACION
vi /etc/kibana/kibana.yml
    # Modificar con el nombre del host específico si se
    #quiere acceder por un nombre que no sea localhost
    server.host: "node-1"

    # Nombre que se le va a dar al servido
    server.name: "node-1"

    # Url's de los nodos de Elasticsearch
    elasticsearch.hosts: ["https://node-1:9200", "https://node-2:9200", "https://node-3:9200"]

    # Modificar el usuario de acceso
    elasticsearch.username: "kibana_system"
    elasticsearch.password: "XXXX"

mkdir /data/kibana

/usr/share/elasticsearch/bin/elasticsearch-certutil csr -name kibana-server -dns example.com,www.example.com
```

```
25
26 scp elasticsearch-ca.pem node-2:/etc/kibana/
27 scp elasticsearch-ca.pem node-3:/etc/kibana/
28 scp kibana.yml node-3:/etc/kibana/
29 scp kibana.yml node-2:/etc/kibana/
30
31 # LEVANTAR SERVICIO
32 systemctl start kibana
33 systemctl stop kibana
34 tail -f /var/log/kibana/kibana.log -n 100
```

7.1.3. Apache Nifi

```
1  # INSTALACION APACHE NIFI 1.22.0
2  wget https://dlcdn.apache.org/nifi/1.22.0/nifi-1.22.0-bin.zip
3
4  yum install -y unzip
5  unzip nifi-1.22.0-bin.zip
6
7
8  # CONFIGURACION
9  vi nifi-1.22.0/conf/nifi.properties
10
11     # Configurar en modo cluster
12     nifi.cluster.is.node=true
13     nifi.cluster.node.address=localhost
14     nifi.cluster.node.protocol.port=5888
15
16     # Ańdir clave de seguridad común para todos los nodos
17     nifi.sensitive.props.key=XXXX
18
19     # Indicar el string de conexión al zookeeper
20     nifi.zookeeper.connect.string=node-1:2181,node-2:2181,node-3:2181
21
22     # Indicar el hostname de la web y el puerto
23     nifi.web.https.host=localhost
24     nifi.web.https.port=443
25
26     # Configurar SSL
27     nifi.cluster.protocol.is.secure=true
28
29 vi /data/nifi-1.22.0/conf/state-management.xml
30
```

```
    <cluster-provider>
            <id>zk-provider</id>
            <class>org.apache.nifi.controller.state.providers.zookeeper.
                ZooKeeperStateProvider</class>
            <property name="Connect String">node-1:2181,node-2:2181,node-3:2181</property
                >
            <property name="Root Node">/nifi</property>
            <property name="Session Timeout">10 seconds</property>
            <property name="Access Control">Open</property>
    </cluster-provider>

vi /data/nifi-1.22.0/conf/login-identity-providers.xml
<provider>
        <identifier>single-user-provider</identifier>
        <class>org.apache.nifi.authentication.single.user.SingleUserLoginIdentityProvider
            </class>
        <property name="Username">user</property>
        <property name="Password">XXXX</property>
</provider>

# GENERAR CLAVES
./bin/tls-toolkit.sh standalone -d 1085 -n node-1,node-2,node-3
nifi-toolkit-1.23.0/bin/tls-toolkit.sh standalone -d 1085 -n node-1,node-2,node-3 -f nifi
    .properties -O -P XXXX -S XXXX

# INCIAR SERVICIO
/data/nifi-1.22.0/bin/nifi.sh start
tail -n 1000 /data/nifi-1.22.0/logs/nifi-app.log -f
```

7.1.4. Apache Zookeeper

```
# INSTALACION APACHE ZOOKEEPER 3.8.2
yum install wget tar
wget https://dlcdn.apache.org/zookeeper/zookeeper-3.8.2/apache-zookeeper-3.8.2-bin.tar.gz
tar -xvf apache-zookeeper-3.8.2-bin.tar.gz

# CONFIGURACION
vi apache-zookeeper-3.8.2-bin/conf/zoo.cfg
        server.1=node-1:2888:3888
        server.2=node-2:2888:3888
        server.3=node-3:2888:3888
mkdir /tmp/zookeeper

```

```
vi /tmp/zookeeper/myid
        1 (diferente para cada instancia)

# LEVANTAR SERVICIO
/root/apache-zookeeper-3.8.2-bin/bin/zkServer.sh start
/root/apache-zookeeper-3.8.2-bin/bin/zkServer.sh status
```

7.1.5. Apache Kafka

```
# INSTALAR KAFKA 3.5.0
wget https://downloads.apache.org/kafka/3.5.0/kafka-3.5.0-src.tgz
tar -xvf kafka-3.5.0-src.tgz

# CONFIGURACION
vi kafka_2.12-3.5.0/config/server.properties
    # Indicar un id al broker (uno distinto por cada nodo)
    broker.id=0

    # Indicar los puntos de conexión a los cuales el servidor Kafka estará escuchando
    listeners=PLAINTEXT://node-1:9092

    # Indicar ruta en la que se escribirán los datos
    log.dirs=/data/kafka-logs

    # Indicar string de conexión a Zookeeper
    zookeeper.connect=node-1:2181,node-2:2181,node-3:2181


## CREAR SERVICIO

vi /etc/systemd/system/kafka.service

[Unit]
Description=Apache Kafka
After=network.target

[Service]
Type=simple
User=root
ExecStart=/kafka-3.5.0/bin/kafka-server-start.sh /kafka-3.5.0/config/server.properties
Restart=always
RestartSec=5

```

```
[Install]
WantedBy=multi-user.target

# ARRANCAR SERVICIO
systemctl daemon-reload
systemctl start kafka
systemctl status kafka
systemctl enable kafka
tail /kafka_2.12-3.5.0/logs/server.log

# CONSUMIR Y PRODUCIR
kafka_2.12-3.5.0/bin/kafka-topics.sh --list --bootstrap-server node-1:9092,node-2:9092,
    node-3:9092
kafka_2.12-3.5.0/bin/kafka-console-producer.sh --broker-list node-1:9092,node-2:9092,node
    -3:9092 --topic topic-name
/data/kafka_2.12-3.5.0/bin/kafka-console-consumer.sh --bootstrap-server node-1:9092,node
    -2:9092,node-3:9092 --topic topic-name --from-beginning
```

7.2. Modelos generados

7.2.1. Redes Neuronales (word embedding + LSTM)

```
from scikeras.wrappers import KerasClassifier
from sklearn.model_selection import GridSearchCV
from keras import optimizers

from tensorflow.keras.callbacks import EarlyStopping

def create_model( embedding_vec_length=300, units=300, learning_rate=0.01):
    print("-----------")
    print("embedding_vec_length", embedding_vec_length)
    print("units", units)
    print("learning_rate", learning_rate)
    print("-----------")
    max_text_length=30
    embedding_layer = Embedding(len(tokenizer.word_index) + 1,
                                embedding_vec_length,
                                input_length=max_text_length,
                                trainable=False)
```

```
    model = Sequential()
    model.add(embedding_layer)
    model.add(LSTM(units))

    model.add(Dense(num_classes, activation='softmax'))
    adam = optimizers.Adam(learning_rate=learning_rate)
    model.compile(optimizer=adam, loss='categorical_crossentropy')
    return model

# Create a KerasClassifier with the model building function
model = KerasClassifier(build_fn=create_model, epochs=50, batch_size=512,
                        embedding_vec_length=300,
                        learning_rate=0.1,
                        units =300,
                        #max_text_length =[10, 20, 30]
                        )
model.get_params().keys()
# Define the hyperparameters to search
param_grid = {
    'embedding_vec_length': [100, 300, 400],
    #'max_text_length': [10, 20, 28],
    'units': [200, 300],
    'learning_rate': [0.01, 0.1],
}

# Create GridSearchCV
early_stopping = EarlyStopping(monitor='val_loss', patience=3,
    restore_best_weights=True)

grid = GridSearchCV(estimator=model, param_grid=param_grid, cv=3, error_score="
    raise")
grid_result = grid.fit(X_train,
                       y_train_one_hot,
                       validation_split=0.2,  # Use a validation split
                       callbacks=[early_stopping])  # X_train and y_train are
                           your training data
```

El mejor modelo fue 7.1:

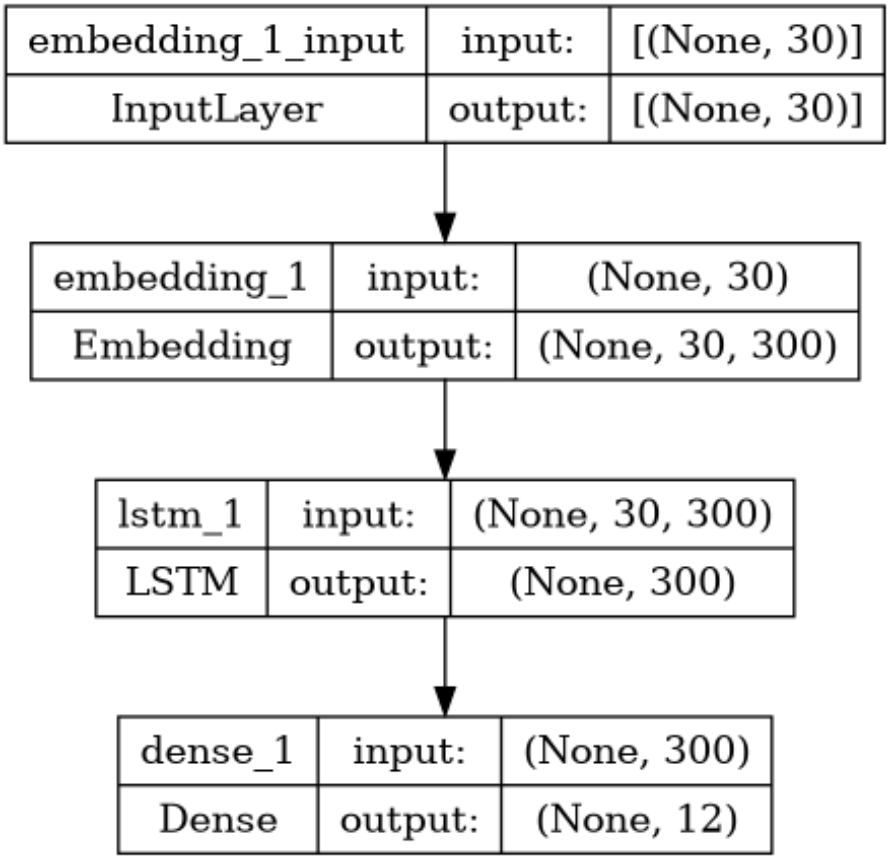

Figura 7.1: Modelo de word embedding + LSTM.

```
from keras.models import Sequential
from keras.layers import Dense, LSTM, Embedding, RepeatVector
from keras.utils import plot_model
from keras.layers import Embedding
from keras import optimizers


embedding_vec_length = 300
max_text_length = 30
units = 300
learning_rate= 0.01
embedding_layer = Embedding(len(tokenizer.word_index) + 1,
                            embedding_vec_length,
                            weights=[embedding_matrix],
                            input_length=max_text_length,
                            trainable=False,
                            mask_zero=True)

def define_model(embedding_layer,units, out_vocab_size):
    model = Sequential()
    # Embedding
    model.add(embedding_layer)

    # LSTM
    model.add(LSTM(units))

    # Output
    model.add(Dense(out_vocab_size, activation='softmax'))
    return model

model = define_model(embedding_layer, units, num_classes)

adam = optimizers.Adam(learning_rate=0.01)
model.compile(optimizer=adam, loss='categorical_crossentropy')

plot_model(model, show_shapes=True, show_layer_names=True)
```

7.2.2. Regresión Lineal

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

# Create logistic regression model
model = LogisticRegression()

# Define hyperparameters to tune
param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'solver': ['liblinear', 'newton-cg', 'lbfgs', 'sag', 'saga'],
    'penalty': ['l1', 'l2', 'elasticnet', 'none'],
    'class_weight': [None, 'balanced']
}

# Use GridSearchCV to find the best hyperparameters
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, scoring='
    accuracy', cv=5)
grid_search.fit(Train_X_Tfidf, y_train_encoded)

# Print the best hyperparameters
print("Best Hyperparameters:", grid_search.best_params_)

# Get the best model
best_model = grid_search.best_estimator_

# Evaluate the model on the validation set
accuracy = best_model.score(X_val, y_val)
print("Validation Accuracy:", accuracy)
# Print the best parameters and corresponding accuracy
print("Best: %f using %s" % (grid_search.best_score_, grid_search.best_params_))
```

El mejor modelo fue:

```
# Assuming you have a trained logistic regression model named 'logreg_model'
logreg_model = LogisticRegression(C=1, class_weight='balanced', penalty= 'l2',
    solver= 'liblinear')

# Assuming X_train and y_train are your training data
```

```
logreg_model.fit(Train_X_Tfidf, y_train_encoded)
```

7.2.3. Random Forest

```
# Create a Random Forest classifier
rf_model = RandomForestClassifier()

# Define the parameter grid to search
param_grid = {
    'n_estimators': [10, 20, 50],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create GridSearchCV
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=3,
   scoring='accuracy')

# Fit the model to the training data
grid_search.fit(Train_X_Tfidf, y_train_encoded)

# Get the best parameters and best model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
```

El mejor modelo fue:

```
# Assuming X_train and y_train_encoded are your training data
rf_model = RandomForestClassifier(max_depth = None,
                                  min_samples_leaf = 1,
                                  min_samples_split = 10,
                                  n_estimators=50,
                                  random_state=42)
rf_model.fit(Train_X_Tfidf, y_train_encoded)

# Assuming X_test is your test data
y_pred_encoded = rf_model.predict(Test_X_Tfidf)
```

7.2.4. Support Vector Machine

```
# Define the SVM model
svm_model = SVC()

# Define the parameter grid to search
param_grid = {
    'C': [0.1, 1, 10],              # Regularization parameter
    'kernel': ['linear', 'rbf'],    # Kernel type
    'gamma': ['scale', 'auto'],     # Kernel coefficient for 'rbf'
}

# Create GridSearchCV
grid_search = GridSearchCV(estimator=svm_model,
                           param_grid=param_grid,
                           cv=3,
                           scoring='accuracy',
                           verbose=10)

# Fit the model to the training data
grid_search.fit(Train_X_Tfidf, y_train_encoded)

# Get the best parameters and best model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
```

El mejor modelo fue:

```
SVM = SVC(C=1, kernel='rbf', gamma='scale')
SVM.fit(Train_X_Tfidf, y_train_encoded)
```