Análisis, selección e implementación de tecnologías de observabilidad y supervisión continua

Luis Carlos Rendón Cardona

Informe de práctica para optar al título de Ingeniero de Sistemas

Semestre de Industria

Asesora

Deisy Loaiza Berrío, Ingeniera de sistemas

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería de Sistemas

Medellín, Antioquia, Colombia

2024

Análisis, selección e implementación de tecnologías de observabilidad y supervisión continua

Cita	Rendón Cardona [1]
Referencia Estilo IEEE (2020)	[1] Rendón Cardona, L.C, "Análisis, selección e implementación de tecnologías de observabilidad y supervisión continua", Semestre de industria, Ingeniería de sistemas, Universidad de Antioquia, Medellín, 2024.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

API	Application Programming Interface (Interfaz de programación de aplicaciones)
JSON	JavaScript Object Notation (Notación de Objetos de JavaScript)
IOC	Indicator of Compromise (Indicador de Compromiso)
DAR	Dificultad, Afectación y Riesgo
UEBA	User and Entity Behavior Analytics (Analítica de Comportamiento de Usuarios y Entidades)
KPI	Key performance indicator (indicador clave de rendimiento)
ELK	Elasticsearch, Logstash, Kibana
HTTP	Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto)
REST	Representational State Transfer (Transferencia de Estado Representacional)

RESUMEN

Business Process Technologies(BPT) empresa de soluciones tecnológicas necesitaba realizar el análisis y la selección de una herramienta de supervisión continua que permitiera realizar el seguimiento constante de uno de los aplicativos desarrollados para uno de sus clientes, partiendo del hecho de que es un tema de vital importancia para la industria en la actualidad y su implementación en proyectos sea en estado de producción o en fase de desarrollo facilita en gran medida dar mantenibilidad a la solución desarrollada gracias a estos seguimientos podría tener una detección temprana de errores dentro de este aplicativo y así dar una solución óptima y eficaz, además sería la base para la implementación en otros proyectos de software o en cualquier tipo de solución tecnológica, se realizó una investigación exhaustiva en donde se partió de unos flujos de mantenimiento seguidos de estrategias de supervisión continua y observabilidad con un conjunto de pasos a seguir que llegaron a facilitar mucho el trabajo y a servir como intermediarios sin limitar la implementación de una solución a una sola tecnología sino permitiendo expandir los horizontes ante cualquier otra, se hizo entonces una selección de una de las herramientas previamente investigadas y se logró instalar y probar de manera exitosa en un entorno local debido a que surgieron novedades por parte de los clientes por lo que hubo de posponerse su posterior instalación en un entorno productivo, pero las pruebas en entorno local fueron totalmente exitosas por lo que su despliegue a producción no significaría un trabajo muy extenso sino por el contrario algo más sencillo gracias a los avances que lograron y que se exponen en este informe.

Palabras clave — **Supervisión, métricas, monitoreo, observabilidad, seguimiento, bugs, rendimiento, datos, información, almacenamiento**

ABSTRACT

Business Process Technologies (BPT), a technological solutions company, needed to carry out the analysis and selection of a continuous monitoring tool that would allow constant monitoring of one of the applications developed for one of its clients, based on the fact that it is a matter of vital importance for the industry today and its implementation in projects, whether in production or in the development phase, greatly facilitates providing maintainability to the solution developed thanks to these monitoring, it could have early detection of errors within this application and thus provide an optimal and effective solution, it would also be the basis for implementation in other software projects or in any type of technological solution, an exhaustive investigation was carried out starting from maintenance flows followed by continuous supervision and observability strategies with a set of steps to follow that made the work much easier and served as intermediaries without limiting the implementation of a solution to a single technology but rather allowing the horizons to be expanded before any other, a selection was then made of one of the previously researched tools and It was successfully installed and tested in a local environment due to new developments from customers, so its subsequent installation in a productive environment had to be postponed, but the tests in the local environment were completely successful, so its deployment Production would not mean a very extensive job but on the contrary something simpler thanks to the advances they achieved and which are set out in this report.

Keywords — **Supervision, metrics, monitoring, observability, tracking, bugs, performance, data, information, storage**

1. INTRODUCCIÓN

Business Process Technologies(BPT) es una empresa de soluciones tecnológicas para organizaciones, responsable de desarrollar productos innovadores y de alta calidad basados en los modelos de negocio de sus clientes, partiendo principalmente del análisis de negocio, la arquitectura y la experiencia de usuario en etapas tempranas. construir el producto necesario de forma ágil e interactiva, aplicando las mejores prácticas y aportando valor agregado constantemente a la organización.

Actualmente BPT se encuentra realizando labores de mejora continua y desarrollo a un aplicativo que realizó hace 1 año a uno de sus clientes, este aplicativo se encuentra actualmente en producción, por lo que las labores de desarrollo van más orientadas a su mejora y evolución constante. La supervisión de este aplicativo es un tema bastante fundamental puesto que cualquier error o vulnerabilidad del mismo puede significar una perdida para esta empresa bien sea de clientes o de dinero por lo que no se le puede dar cabida a ningún tipo de error.

Es importante tener en cuenta el poco tiempo que lleva el aplicativo en producción y si bien no han ocurrido temas que pasen a mayores, si han ocurrido algunas fallas o bugs que se pudieron haber evitado si hubiera de alguna forma algún vigía constante o algo que alertara de forma temprana y además de una manera concisa y fácil de observar cualquier tipo de error.

BPT se encargará de brindar una solución a esta problemática de supervisión continua y lo hará por medio de la implementación de la tecnología que mejor se adapte al tipo de tecnología y la arquitectura con la que funcione el programa al que se le realizará la supervisión.

2. PLANTEAMIENTO DEL PROBLEMA

A pesar de los esfuerzos de BPT por mejorar continuamente el aplicativo para su cliente, se enfrentan a un desafío significativo relacionado con la detección temprana de errores y vulnerabilidades en el sistema. Aunque el aplicativo ha estado en producción por solo un año, ya han surgido algunas fallas y bugs que podrían haberse evitado con una supervisión más constante y efectiva.

El problema principal es la falta de un sistema de vigilancia proactivo que alerte de manera inmediata y clara sobre cualquier anomalía en el funcionamiento del aplicativo. Esto ha llevado a situaciones donde los errores pasan desapercibidos durante un tiempo considerable, lo que puede provocar pérdidas tanto para BPT como para su cliente, ya sea en términos de reputación o incluso financieros.

Además, la ausencia de un mecanismo de supervisión eficaz ha generado una sensación de incertidumbre tanto en el equipo de desarrollo de BPT como en el cliente. Ambas partes están preocupadas por la posibilidad de que problemas no detectados puedan surgir en el futuro y causar un impacto aún mayor.

3. OBJETIVOS

3.1. Objetivo general

Implementar la pila ELK (Elasticsearch, Logstash, Kibana) en el entorno productivo de una aplicación para permitir una mejora continua y análisis exhaustivo de todos los registros generados por la misma, facilitando así un monitoreo efectivo y la detección temprana de errores.

3.2. Objetivos específicos

- Desplegar Elasticsearch como motor de búsqueda.
- Configurar Logstash para la ingestión de datos.
- Establecer Kibana como interfaz de usuario para la visualización de métricas establecidas.
- Desarrollar paneles de control personalizados en Kibana

4. MARCO TEÓRICO

La supervisión continua es una actividad de gran importancia para facilitar a las organizaciones la mantenibilidad de cualquier tipo de producto y la identificación temprana de fallos mucho antes de que puedan evolucionar a una escala mayor de la que actualmente están. El constante seguimiento y monitoreo de estos datos favorece a los distintos proyectos para saber que tan bien o mal se ejecutan los procesos internos de un aplicativo, gracias a ellos tanto desarrollador como cliente (si es el caso), se pueden favorecer de una descripción detallada de dichos procesos y una identificación temprana de errores si existen u ocurren en un momento determinado.

"Conocer el comportamiento de los fallos, cómo se manifiestan y su curva de ocurrencia es fundamental para crear estrategias dentro de un plan de mantenimiento." [1] cabe resaltar que es de vital importancia entonces tener una visión clara sobre las posibles ocurrencias de errores en cualquier tipo de proyecto para detectar patrones que ayuden a dar con una solución óptima de los problemas que se presentan y esto solo se puede lograr gracias a la detección temprana de estos errores y gracias a la visión explícita del como se pudo haber producido el fallo.

Para nadie es un secreto que de los errores se aprende y es algo que no es una excepción para el mundo del software, a lo largo de la historia se han podido evidenciar muchos casos de plataformas tecnológicas que sucumbieron ante la carencia de una evolución constante y una mejora continua, un ejemplo muy claro puede ser Windows vista que tuvo un rotundo fracaso en el mercado debido a muchas razones, pero una de ellas fue la falta de supervisión y mejora constante; si hubiera existido una mayor atención a la calidad, interacción y usabilidad este fracaso fuera su fin.

"La falta de supervisión o la inexistencia de mantenimiento del sistema degeneraría en el desempeño inesperado y sus resultados generarían desconfianza en el entorno" [2]; es por esto que las organizaciones deben asegurar de alguna manera su productividad y garantizar la misma solo depende de que tan bien funcionan sus herramientas de trabajo y más aún si su punto fuerte depende en gran medida de una herramienta de software que puede llegar a presentar fallos ya sea por la evolución del entorno en el que está trabajando o si alguna de sus características debe evolucionar o cambiar para adaptarse a nuevas reglas.

Como bien es dicho en el artículo sobre detección temprana de errores: "En el mundo del desarrollo de software, la detección y corrección oportuna de errores (bugs) son fundamentales para garantizar la estabilidad y la funcionalidad de las aplicaciones y programas" [3]. Con el continuo avance de la tecnología y la creciente complejidad de los sistemas informáticos, las pruebas automatizadas

han surgido como una herramienta indispensable en la detección temprana de bugs. La supervisión continua es entonces una garantía de la alta disponibilidad y la confiabilidad que puede tener un software ya sea en etapas tempranas al realizar constantes pruebas o en etapas finales para realizar el fortalecimiento de los procesos internos que este posee.

Es en este punto entonces en donde se debe dar una descripción del aplicativo que actualmente requiere esa constante supervisión y del por qué es tan indispensable destinar una herramienta automatizada para que ayude a obtener métricas y visualización de su patrón de comportamientos para detectar errores. Como se mencionó antes, BPT se encuentra en el actual desarrollo de nuevas funcionalidades para un aplicativo que desarrolló hace 1 año para uno de sus clientes y que se encuentra actualmente en su fase de producción, BPT también se encarga de realizar labores de mantenimiento ante cualquier bug que el aplicativo llegue a presentar.

El aplicativo en cuestión cuenta con un punto de venta, una aplicación central, una API y una web y su función principal es ayudar con la gestión de ventas de 25 almacenes, así como de su administración y correcta sincronización.

Cuando un aplicativo está en producción lo ideal es establecer un flujo de mantenimiento que se puede dividir en varios tipos:

- **Correctivo**: Se centra en corregir errores o defectos en el software identificados después de su implementación.
- **Adaptativo**: Implica modificar el software para adaptarse a cambios en el entorno operativo, como actualizaciones de hardware o software de terceros.
- **Perfectivo**: Se concentra en mejorar el rendimiento o la usabilidad del software sin cambiar su funcionalidad principal.
- **Preventivo**: Busca evitar problemas futuros mediante la identificación y corrección de posibles fallos antes de que ocurran.

Todos ellos adecuados y necesarios para que un producto de software pueda mejorar su estado de rendimiento, pero para que se puedan dar todos estos tipos de mantenimiento y para que se puedan llegar a establecer las métricas mencionadas es necesario tener algo que permita realizar un seguimiento, algo que se pueda observar ya que un producto de software cuenta con muchas características que pueden llegar a ser medidas y observadas para establecer patrones que ayuden a detectar errores y prevenirlos, por ejemplo el código escrito, la infraestructura, los pipelines o los

logs del aplicativo y actualmente existen múltiples tecnologías que son de gran ayuda para poder realizar labores de supervisión y observabilidad como los que se listaran a continuación.

4.1. Azure Monitor.

Imagen número 1(Azure)

Azure Monitor es una herramienta integral de supervisión proporcionada por Microsoft Azure. Su objetivo es ayudarte a supervisar, solucionar problemas y optimizar tus aplicaciones y recursos en la nube y en entornos locales. Azure Monitor recopila datos de diversas fuentes, como máquinas virtuales, aplicaciones, servicios, contenedores y más. Estos datos incluyen métricas, registros y trazas, además es posible configurar alertas basadas en umbrales para recibir notificaciones cuando se produzcan eventos inesperados. Posee herramientas de consumo como lo son:

- Azure Portal: Es la interfaz gráfica principal para interactuar con Azure Monitor[4].
- Azure Monitor para contenedores: Proporciona métricas específicas para contenedores Docker y Kubernetes[4].
- Azure Monitor para aplicaciones: Incluye Application Insights, que se centra en supervisar aplicaciones web y servicios[4].
- Azure Monitor para VM: Supervisa máquinas virtuales y recopila métricas y registros[4].

4.2. Prometeus

es un sistema de monitoreo de código abierto que se utiliza para recopilar y almacenar datos de series temporales. Su objetivo principal es supervisar aplicaciones y sistemas, y proporciona una forma eficiente de recopilar métricas y eventos. Algunos aspectos clave de Prometheus son:

- Prometheus consta de varios componentes, incluido el servidor Prometheus y los exportadores [5]..
- El servidor Prometheus recopila datos de los exportadores y almacena las métricas en su base de datos interna [5].
- Los exportadores son agentes que se ejecutan en los hosts que se desea monitorear. Exponen métricas específicas del sistema o de aplicaciones [5].

4.3. Grafana

Por otro lado, Grafana es una plataforma de visualización de datos de código abierto que se integra muy bien con Prometheus y otros sistemas de monitoreo. Con Grafana, los usuarios pueden crear paneles y gráficos personalizados para representar sus datos de manera visual y comprensible. Grafana ofrece una amplia gama de opciones de visualización, desde simples gráficos de líneas hasta complejas visualizaciones de datos en tiempo real, lo que la convierte en una herramienta poderosa para el análisis y la monitorización de sistemas. Además, Grafana permite la creación de alertas basadas en los datos recopilados, lo que proporciona a los usuarios la capacidad de responder rápidamente a cualquier problema que pueda surgir. En resumen, Prometheus y Grafana son herramientas complementarias que juntas ofrecen una solución completa para el monitoreo, la visualización y la gestión de sistemas y servicios en entornos informáticos modernos[6].

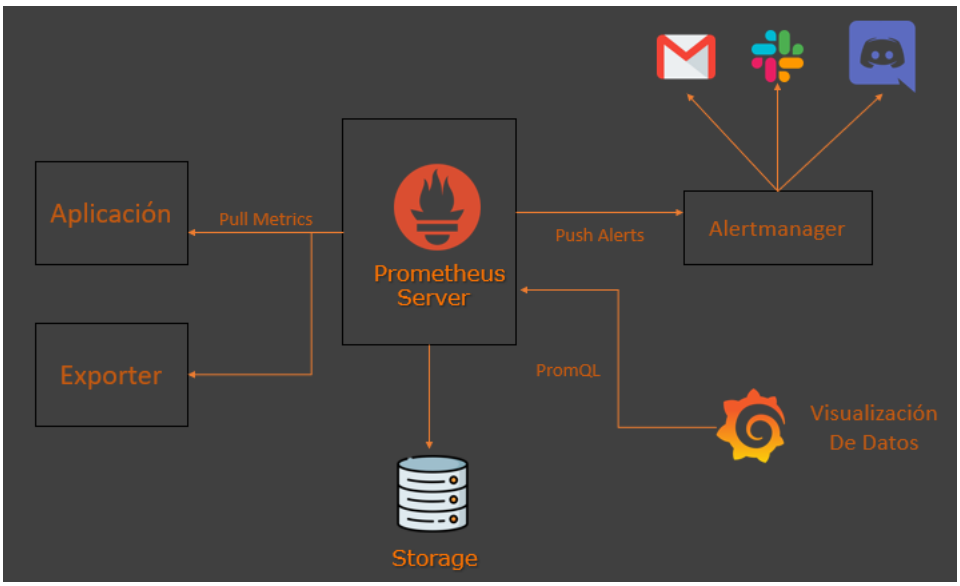

Imagen número 2(Arquitectura Grafana y Prometeus)[7]

4.4. ELK Stack

Es la combinación de Elastic Search, Logstash y Kibana que se utiliza para proporcionar un enfoque integral en la consolidación, gestión y análisis de registros de las aplicaciones [8].

- **Elasticsearch:** Es un motor de búsqueda y análisis distribuido que almacena y busca datos en tiempo real. Su capacidad para indexar y buscar grandes volúmenes de información lo convierte en una herramienta valiosa para el monitoreo.
- **Logstash:** Se encarga de la ingesta y monitoreo de datos, su función principal es detectar los cambios y proporcionar esta información a ElasticSearch.
- **Kibana:** Proporciona una interfaz de usuario visual para explorar, visualizar y analizar los datos almacenados en Elasticsearch. Con Kibana, puedes crear dashboards personalizados y visualizaciones para obtener insights rápidos.

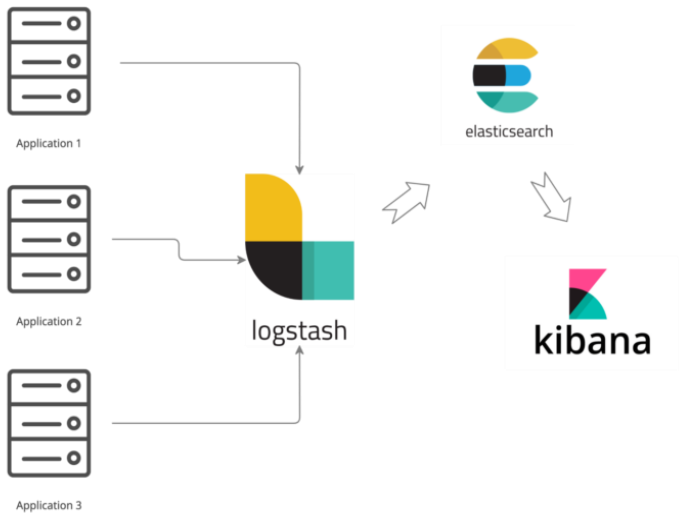

Imagen número 3(Arquitectura de la pila ELK Stack)[5]

4.5. Graylog

Es una plataforma de gestión de registros de código abierto diseñada para recopilar, indexar y analizar grandes volúmenes de registros de diferentes fuentes en tiempo real. Proporciona capacidades de búsqueda, correlación y visualización de registros para ayudar a los equipos de operaciones de TI a identificar y solucionar problemas en sistemas distribuidos y entornos de infraestructura complejos.

La plataforma Graylog está construida sobre Elasticsearch para el almacenamiento y la indexación de datos, MongoDB para la persistencia de metadatos y configuraciones, y utiliza un servidor de aplicaciones Java para proporcionar una interfaz de usuario web. Graylog es altamente escalable y se puede implementar en entornos de cualquier tamaño, desde pequeñas empresas hasta grandes organizaciones empresariales [9].

- Recopilación de registros: Permite recopilar registros de una variedad de fuentes, como servidores, aplicaciones, dispositivos de red y más.
- Análisis de registros en tiempo real: Ofrece capacidades de búsqueda y filtrado en tiempo real para ayudar a identificar problemas de manera rápida y eficiente.
- Correlación de eventos: Permite correlacionar eventos de diferentes fuentes para detectar patrones y anomalías.
- Visualización de datos: Proporciona paneles personalizables y gráficos para visualizar datos de registros de manera efectiva.
- Alertas: Permite configurar alertas basadas en eventos específicos para notificar a los usuarios sobre problemas críticos o situaciones anómalas.

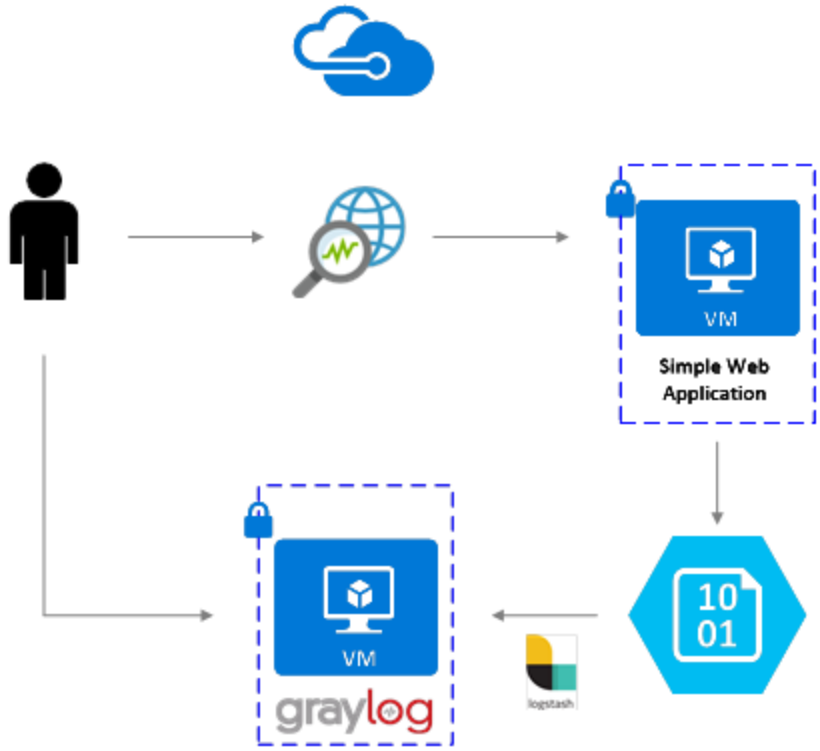

Imagen número 4(Arquitectura Graylog) [9]

4.6. Splunk

Es una plataforma líder en el mercado para la gestión y análisis de datos operativos. Se utiliza para buscar, monitorear, visualizar y analizar grandes volúmenes de datos generados por aplicaciones, sistemas y dispositivos en tiempo real. A través de su arquitectura altamente escalable y su amplia gama de características, Splunk permite a las organizaciones obtener insights valiosos y tomar decisiones informadas basadas en los datos, puede recopilar datos desde una amplia variedad de fuentes, incluidos registros de servidores, eventos de seguridad, métricas de sistemas, datos de sensores IoT y más[10].

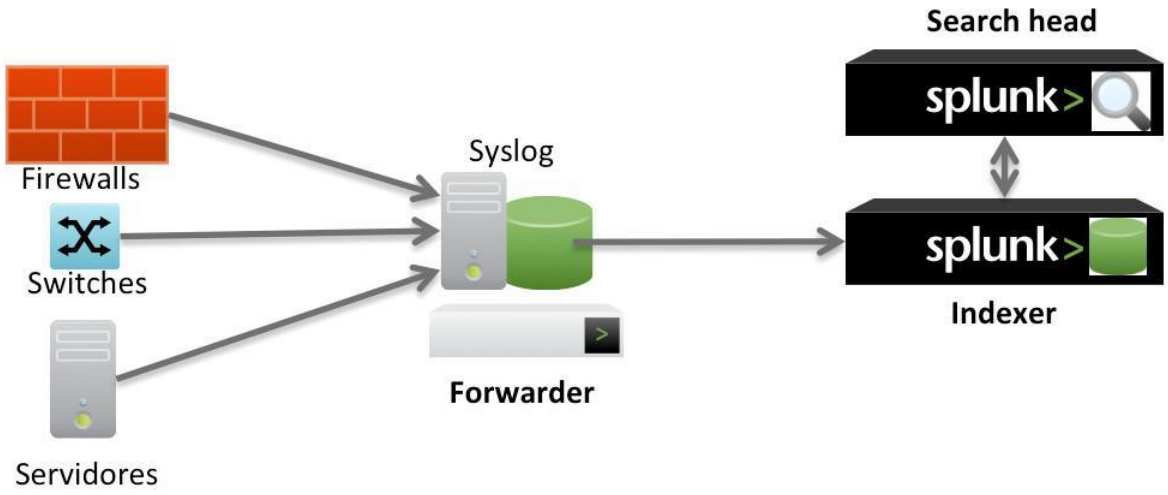

Imagen número 5(Arquitectura Splunk) [11]

Adicional a las herramientas expuestas anteriormente existen muchas otras que pueden ser de gran utilidad en el momento de realizar una observabilidad, pero según la investigación hecha fueron las que mejor se llegaron a ajustar a los tipos de requerimientos, sin embargo, se hizo un análisis para determinar cuál era la más opcionada según ciertos criterios de aceptación indispensables para poder seleccionar la herramienta que mejor se adapte y se puede apreciar en la tabla número 1

Tecnología	Escalabilidad	Usabilidad	Flexibilidad	Visualización	Comunidad activa
ELK	Alta	Moderada	Alta	Alta	Si
Azure monitor	Alta	Alta	Alta	Alta	Si
Splunk	Alta	Moderada	Alta	Buena	Si
Prometheus-Grafana	Moderada	Moderada	Alta	Alta	Si
Graylog	Moderada	Moderada	Alta	Buena	Si

Tabla número 1 (Comparación de tecnologías)

Según la investigación realizada todas las tecnologías pueden ser de gran utilidad y todas tienen en común la versatilidad para la recopilación de datos y aportan unas muy buenas opciones para poder realizar una observabilidad, sin embargo según los criterios de aceptación definidos solo 2 de ellas se pueden llegar a ajustar mejor al tipo de software al que se le realizará la labor de observabilidad y estas serían ELK y Azure monitor, pero antes de definirlo hay que tener en cuenta también algunas características importantes también para tomar una decisión.

Algunas de las herramientas que se analizan están encaminadas a ciertos tipos de soluciones que otras o pueden poseer ciertos tipos de características muy importantes para tener en cuenta, Azure monitor por ejemplo es una plataforma más adecuada para aplicaciones en la nube, Splunk es muy potente pero puede llegar a ser muy costoso en cuanto a presupuestos, Prometheus y Grafana puede llegar a ser mucho mejor para los tipos de sistemas basados en contenedores y microservicios, por otro lado ELK puede llegar a ser ideal para manejar grandes volúmenes de datos no estructurados, mientras que Graylog al estar construido sobre Elasticsearch, uno de los componentes principales de ELK puede llegar a ser más limitado en cuanto a las herramientas que ofrece por lo que la mejor alternativa sería utilizar ELK que vendría siendo la herramienta o el grupo de herramientas más adecuado para realizar una excelente gestión de mantenibilidad y supervisión continua y la que se llegaría a ajustar a las necesidades del momento.

Teniendo definida entonces la tecnología más adecuada para realizar las labores de seguimiento y supervisión se lo que se debe definir ahora es una fuente de datos, algo que se pueda observar y que brinde una información medible y que sea fundamental para dar solución a errores además de ayudar también a prevenirlos y el tipo de archivo más adecuado es el log ya que está presente en cada instancia del aplicativo y posee información detallada sobre el funcionamiento del programa, así como la traza completa desde que se inicia su ejecución, tiene información como la fecha, la hora y muestra los detalles de los errores que se presentan. Los logs son tan importantes que en la actualidad los desarrolladores de BPT se guían de ellos para dar solución a errores del aplicativo en cuestión por lo que tenerlos en un solo punto de referencia sería de gran ayuda para el fin deseado que es observar y atacar los posibles problemas.

Si bien la mejor fuente que se puede utilizar para realizar supervisión continua al producto de software elaborado por BPT son los logs, hay que tener en cuenta también que estos deben tener una correcta estructura, que permita que puedan ser leídos con claridad y que así mismo puedan ser entendidos por quienes lo leen de una forma clara y concisa, por lo que su estructura y elaboración es un tema que también se debe tener en cuenta para establecer unas buenas métricas y por ende se incluirá también algunos fragmentos de investigaciones que se consideraron pertinentes para tener en cuenta sea en la implementación que se describirá más adelante o a futuro luego de haber implementado la herramienta de observabilidad, ya que estas prácticas nunca estarían de más sino por el contrario serían un plus en caso de ser tenidas en cuenta.

Si bien todos los productos de software pueden contar con logs no todos tienen una implementación correcta de los mismos, cuando estos se llegan a utilizar con el fin de realizar tareas de observabilidad esta labor exige un poco más de cuidado que el hecho de simplemente mostrar un sinnúmero de información o data no relevante y ocupar espacio no deseado en bases de datos, es por eso que se recomienda usarlos de forma adecuada y para esto existen un conjunto de buenas prácticas como lo son las siguientes:

- **Fecha y hora del log:** Registra el momento exacto en el que ocurrió el evento, lo que es esencial para la trazabilidad y el análisis temporal de los problemas.
- **Nivel del log**: Indica la gravedad del evento registrado, como "FATAL" para errores críticos que detienen el funcionamiento normal del sistema, "WARNING" para advertencias que no son críticas, pero deben ser revisadas, entre otros, en la imagen número 6 a continuación se anexará información esquemática que facilita la toma de decisiones en cuanto a la escogencia del nivel del log.

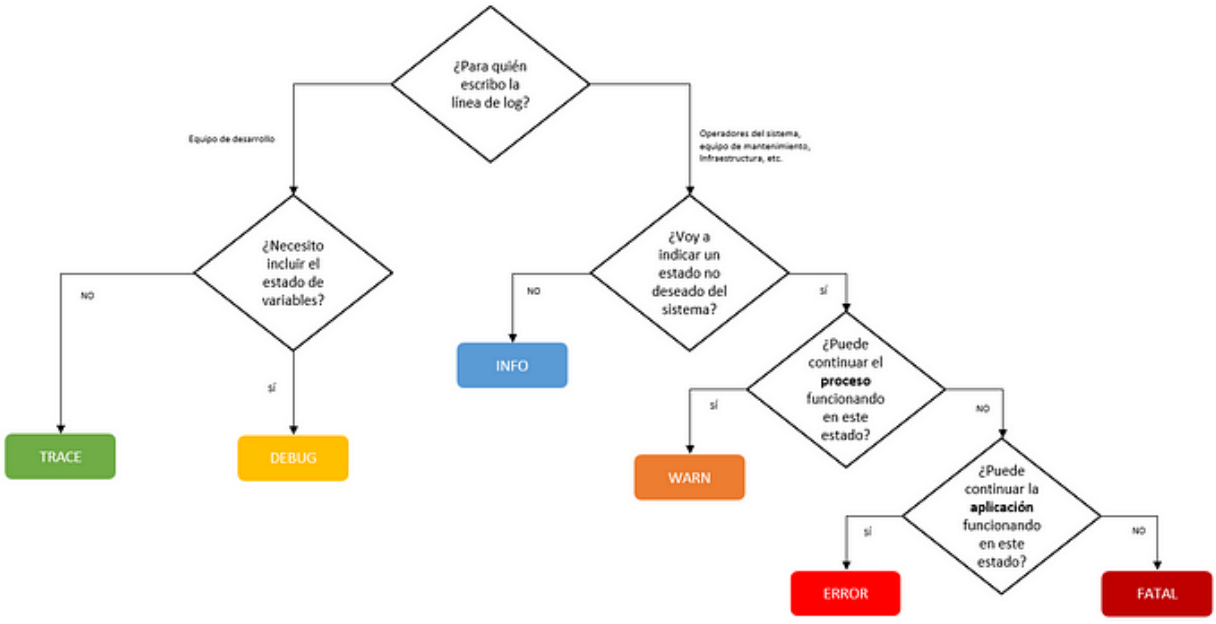

Imagen número 6(Nivel del log) [12]

- **Mensaje:** Proporciona información sobre lo que sucedió, como una descripción del evento o error[13].

- **Stack de la excepción:** Muestra la secuencia de llamadas que llevaron al error, lo que ayuda a identificar la causa raíz del problema[13].
- **Payload:** Contiene los datos enviados con la solicitud HTTP que provocó el error, lo que permite reproducir y entender el contexto en el que se produjo el error[13].
- **Callsite:** Identifica el punto exacto en el código donde ocurrió el error, como el endpoint del API que fue consumido[13].
- **Acción:** Describe la acción específica que estaba en curso cuando se produjo el error, como el nombre del método en el controlador que estaba siendo ejecutado[13].
- **Usuario:** Indica el usuario que inició la acción que desencadenó el error, lo que facilita la identificación de problemas específicos de usuario o de permisos [13].
- **Nombre del método:** Identifica el método específico de la capa de lógica de negocio donde se generó el error, lo que ayuda a localizar y corregir el problema en el código [13].
- **Nombre de la aplicación:** Especifica el nombre de la aplicación en la que se produjo el error, lo que facilita la identificación y el seguimiento de los problemas en sistemas complejos con múltiples aplicaciones [13].
- **Formato de log estándar:** Esto es básico y cualquier librería como log4j o logging en el caso de python facilitan el uso de patrones [12]
- **No incluir información sensible:** Bajo ninguna circunstancia se recomienda registrar contraseñas, datos personales, etc. Ya que en algunos casos podrían facilitar la vida a malwares [12]
- **Escribir logs también para las máquinas:** ELK, DataDog, Dynatrace, etc. está muy extendido el uso de herramientas de observabilidad y monitorización. Si se escriben frases esquemáticas y repetibles se simplificaría el parseo de estos registros [12].

5. METODOLOGÍA

Se utilizó el marco de trabajo ágil basado en SCRUM como metodología principal y para el seguimiento de todas y cada una de las actividades que se programaron para el correcto desarrollo del aplicativo, haciendo más fácil el trabajo en equipo y asegurando el desarrollo de las mejores prácticas para el diseño e implementación de la pila ELK.

Se contó con un equipo conformado por:

- **Desarrolladores:** Personas encargadas de llevar a cabo la implementación de la pila ELK
- **Scrum** master: Facilitará el diálogo y la mediación con el cliente, además de ser quien organiza las daily meeting y los sprints.
- **Arquitecto:** Se encarga de diseñar y desarrollar sistemas de software y en este caso se encargaría de brindar una ayuda oportuna en caso de necesitar asesorías con respecto a la implementación de ELK y a aclarar inquietudes con ciertos conceptos que implican su diseño

Las ceremonias o actividades que se realizaron bajo este marco de trabajo basado en SCRUM fueron:

- **Sprint:** Ejecución de tareas en un tiempo específico de 30 días, donde se exponen las tareas propuestas en el planning
- **Planning:** Esta ceremonia fue realizada en cada sprint y en ella se definieron todas las historias de usuario que se desarrollaron por sprint
- **Daily meeting:** Reunión diaria realizada a las 8:30 donde se expusieron avances, inconvenientes y metas alcanzadas

Metodología de la implementación

El análisis de requisitos implicó estudiar las particularidades de los entornos de producción y las aplicaciones donde estaban involucradas la recopilación de datos, el monitoreo y los registros analíticos. Se determinaron los tipos de datos para almacenar, los formatos de registro, así como los requisitos de visualización y análisis.

Partiendo del análisis hecho aquí sobre qué tecnología sería la más adecuada para trabajar se dirige entonces el uso de ELK como medio para definir las estrategias de observabilidad y se ajustaría a una arquitectura de observabilidad, adaptándola a los puntos clave de esta que son: tener una fuente de datos, gestionar una correcta ingesta de los mismos, definir un punto de almacenamiento para guardar esta información, implementar una plataforma de analítica que permita facilitar la visualización y por último definir métricas que ayuden a identificar patrones en el código para evitar errores y prevenirlos. En el marco teórico se pudo definir cuál era la mejor fuente de datos por lo que se procede a describir los demás puntos a seguir.

Ingesta de datos

Logstash sería herramienta que hace parte de la tecnología ELK para llevar a cabo la tarea de tomar la información de la fuente definida, en este caso los logs del aplicativo generados en cada una de las instancias del programa al que se le realizará el seguimiento de observabilidad, supervisión y mantenimiento, su función principal es ser un listener que detecta cada cambio en la fuente e inmediatamente lo toma y lo manda a Elasticsearch.

Garantizar un punto de almacenamiento.

Elasticsearch

es la pieza central donde se almacena la información en la tecnología ELK, esta herramienta toma todos los servicios enviados por Logstash y los almacena dentro de una estructura de índices que organizan la información para ser leída de una forma fácil y a su vez para que pueda ser filtrada de la manera más óptima.

Implementar una plataforma de analítica

En este caso sería entonces la pieza faltante Kibana, quien se hará cargo de mostrar toda la información recopilada por Logstash y Elasticsearch y brindaría un sinnúmero de herramientas para que la visualización de los índices almacenados se haga efectiva y para que la usabilidad sea un punto clave.

Definir métricas

Kibana por ser la plataforma de analítica ofrece tanto herramientas de visualización como de medición, por ende, sería también la herramienta perfecta para llevar esta labor y ayudar a definir alertas, a medir la información recopilada y a identificar cualquier patrón de comportamiento extraño dentro del código.

Implementación de la solución de observabilidad

Habiendo adaptado entonces la tecnología seleccionada a la arquitectura de observabilidad propuesta lo que se realizó entonces fue la implementación de todas y cada una de las herramientas anteriormente mencionadas dentro de un entorno local, y en este informe solo se anexarán registros de pruebas realizadas en este mismo entorno ya que su despliegue a producción fue pospuesto porque surgieron otro tipo de prioridades a ser desarrolladas por BPT para su cliente, incluyendo labores de mantenimiento del aplicativo y nuevos desarrollos.

6. RESULTADOS

La implementación de estrategias y tecnologías de observabilidad estuvo encaminada a la mejora continua de las características de un aplicativo de software, no es más que una garantía casi segura de un constante crecimiento y un aumento en la robustez del programa, facilitando todos los procesos de solución de bugs y siendo algo clave también para nuevos desarrollos, la etapa de finalización del proceso de implementación de dichas tecnologías de supervisión continua se expone entonces en esta sección del informe, se mostrará a detalle los procesos de instalación, configuración y pruebas realizadas a la implementación de la solución.

Cabe aclarar que la instalación y configuración de ELK pudo realizarse gracias al constante acompañamiento de BPT con la documentación aportada y la respuesta oportuna cuando existieron dudas sobre temas de la arquitectura de monitoreo y observabilidad e información técnica sobre la instalación de las tecnologías implementadas.

El primer paso para la implementación de ELK fue la instalación de Docker para gestionar la instalación las tecnologías necesarias dentro de contenedores que estuvieran ejecutando dichos programas mientras fuese necesario, es importante aclarar que no necesariamente es la única alternativa para la instalación de dichas tecnologías, sino que fue el método escogido durante el proceso por su facilidad en cuanto la implementación.

Teniendo entonces Docker instalado se procedió a realizar la instalación de Elasticsearch y Kibana, siguiendo los tutoriales ofrecidos por BPT y siguiendo también algunos otros encontrados durante el proceso de investigación y análisis, dicha instalación se hizo configurando un archivo Dockerfile en donde se fijó información sobre los puertos en los que se estarían ejecutando estos programas y también el nombre de los contenedores para saber diferenciarlos cuando se estuviesen ejecutando dentro de Docker.

```yaml
version: '3'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.20
    container_name: elastic
    networks:
      - elastic
    ports:
      - "9200:9200"
    environment:
      - discovery.type=single-node
    mem_limit: 2g

  kibana:
    image: docker.elastic.co/kibana/kibana:7.17.20
    container_name: kib01
    networks:
      - elastic
    ports:
      - "5601:5601"

networks:
  elastic:
    driver: bridge
```

Imagen número 7 (Dockefile para instalar Elastic y Kibana)

Se ejecutó entonces el archivo Dockerfile configurado para realizar la instalación y luego al correr las respectivas instancias de los programas desplegados se obtienen los siguientes resultados que no son más que la muestra de que se están ejecutando correctamente.

Imagen número 8 (Dockefile para instalar Elastic y Kibana)

Luego de haber ejecutado los respectivos contenedores se pueden visualizar también en docker para comprobar que funcionan correctamente.

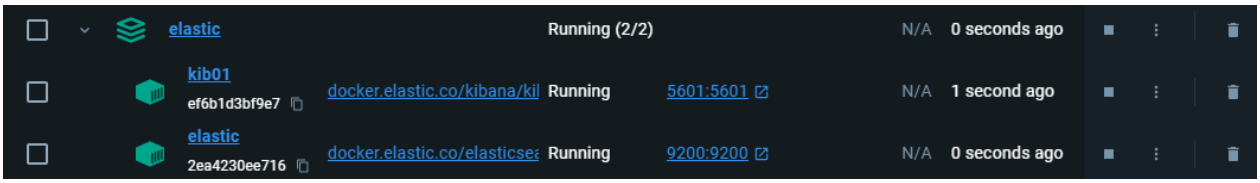

Imagen número 9 (ElasticSeach y Kibana)

Se aclara nuevamente que estas imágenes son las pruebas realizadas en un entorno local, por lo que el objetivo a futuro, justo cuando se pueda realizar el despliegue a producción de dicha implementación de observabilidad sería realizar el mismo proceso, pero en un servidor destinado por el cliente de BPT quienes se encargarían de ponerlo a disposición junto con las características y requerimientos de hardware y software necesarios para que esta instalación sea posible.

Si se ingresa a la ruta: localhost:5601 la primera impresión que se tiene es la de un login que configura por defecto Elasticsearch.

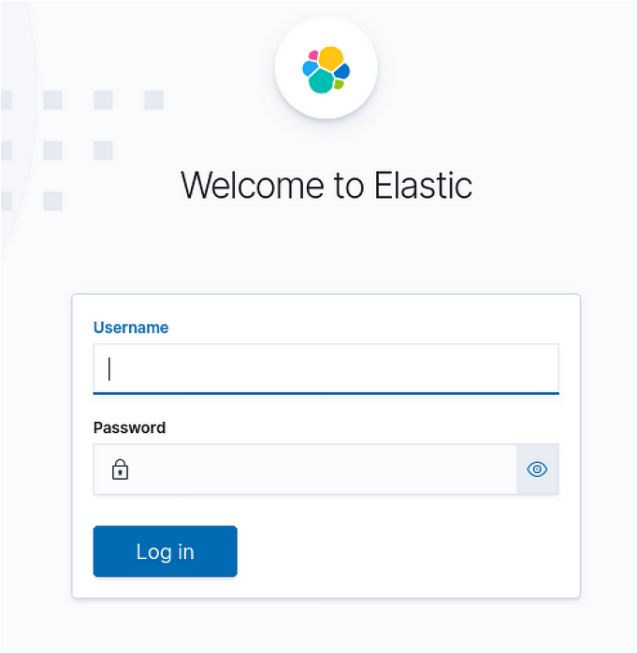

Imagen número 10 (Login Elastic)

Para iniciar sesión solo bastaría ingresar: changeme en usuario y contraseña y se tiene acceso entonces al menú general que ofrece Kibana junto con todo el conjunto de herramientas que posee.

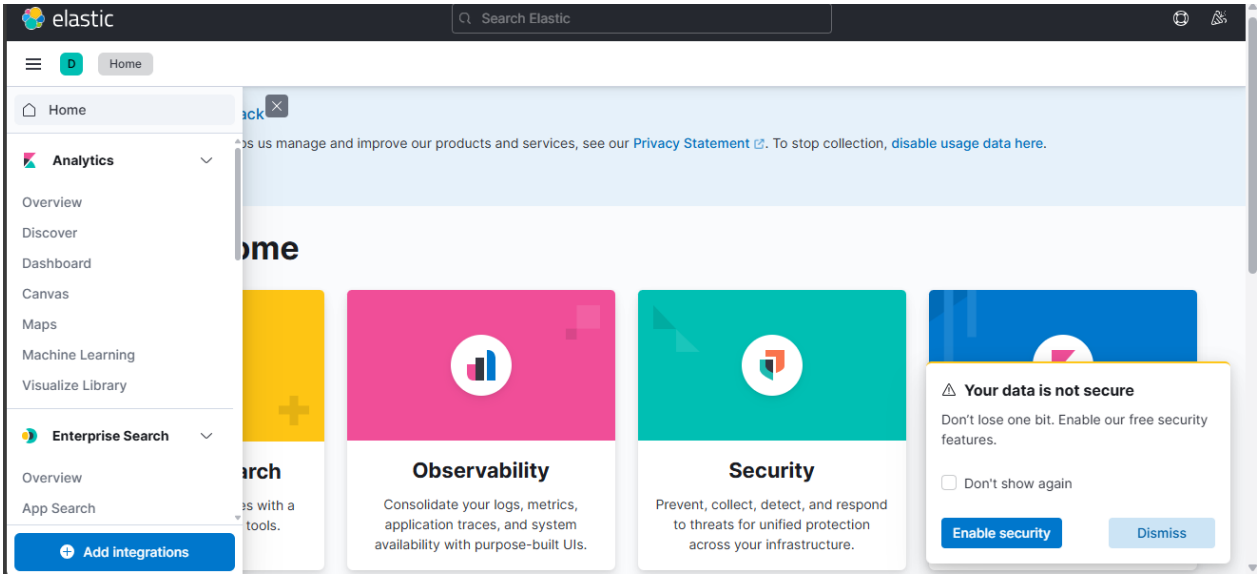

Imagen número 11 (Menú de opciones de Kibana)

Luego de haber realizado la instalación de Kibana y Elastic se procede entonces a instalar logstash, aquí es donde hay que tener algo en cuenta y es que como el cliente de BPT quien posee el software al que se le realizarán los procesos de observabilidad posee un total de 25 almacenes se tendrían entonces todos estos como puntos de información diferentes o aplicaciones en funcionamiento, Logstash se encargaría de estar recopilando esta información y de escucharla en cada punto o aplicativo y cada que se detecte un cambio enviarla a Elasticsearch que haría las veces de base de datos, además de organizar la información y por último se podría visualizar por medio de Kibana que ayudaría también a establecer métricas, balances y gráficas en base a los patrones de comportamientos, también se podrían definir alertas ante la detección de cualquier anomalía y estas anomalías estarían definidas por quien se encargue de instalar y configurar todo. Se estaría entonces centralizando toda la información distribuida en los almacenes en un solo punto y se estaría organizando dicha información para un fácil acceso para quien vaya a dar solución a los posibles errores que ocurran.

Es por esto que la instalación de Logstash no se gestionó por medio de docker quien simula el servidor donde se van a instalar los programas de recopilación y visualización ya que se considera

innecesario que se ejecute junto con estos dos, sino que se ejecutará en estos 25 puntos de referencia que son los almacenes que ejecutan el aplicativo desarrollado por BPT, adicional se estaría ejecutando también en un servidor central que posee un software que gestiona los procesos de sincronización de dicho aplicativo y por último el API que hace parte de dicho programa, siendo así un total de 27 archivos a escuchar y a centralizar en un mismo punto de referencia dentro de Elastic.

Se procedió entonces a hacer la descarga de Logstash dentro de la página oficial de Elastic[14], luego de tener los archivos se procede a abrir el archivo .conf que este contiene.

```
input {
  file {
    path => ["C:/Users/luis9/OneDrive/Documentos/logstash-8.11.1/config/prueba.txt"]
    start_position => "beginning"
  }
}

filter {
  grok {
    match => { "message" => "%{TIMESTAMP_ISO8601:timestamp} %{LOGLEVEL:loglevel} \[%{DATA:thread}\] %{JAVACLASS:logger} - %{GREEDYDATA:message}"
  }
  date {
    match => [ "timestamp", "ISO8601" ]
  }
}

output {
  elasticsearch {
    hosts => ["http://localhost:9200"]
    index => "indice_prueba"
    document_type => "_doc"  # Añade esta línea si estás utilizando Elasticsearch 7.x o superior
  }
}
```

Imagen número 12 (Configuración de Logstash)

Dentro de este archivo de configuración se configuró la estructura de los logs que posee el aplicativo al que se le realizará la observabilidad y la ruta de Elasticsearch, en este caso para el puerto local, pero en un entorno de producción apuntaría a la dirección del servidor, en este archivo también se configuró la ruta del archivo que será leído dentro de la opción index (“índice_prueba”) y luego de esto se procede a realizar la ejecución de Logstash.

Nota: Este último proceso se hace en todos los 25 almacenes, todos estos ejecutados bajo un sistema operativo Windows por lo que al desplegarse en un entorno de producción es necesario programar una tarea para que se ejecuten siempre que se encienda el computador.

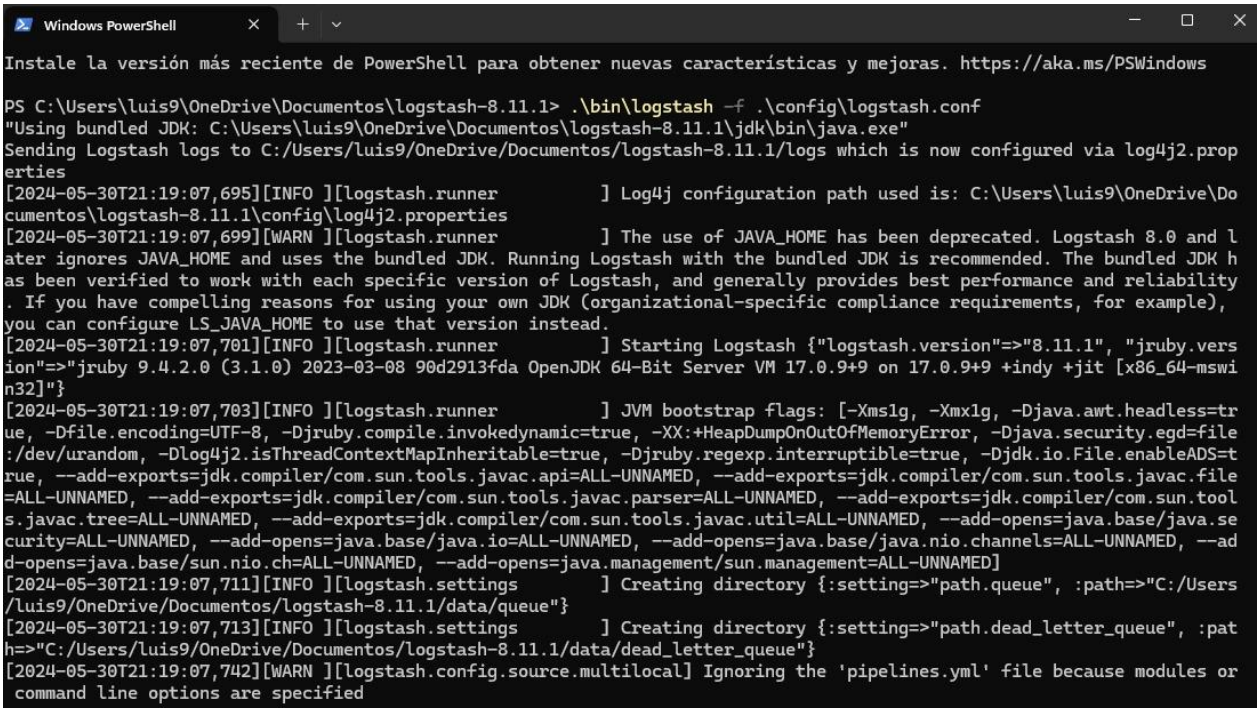

Imagen número 13 (Ejecución de Logstash)

A continuación, se anexa un log de prueba con la información que será leída por Logstash para luego enviar a Elasticsearch

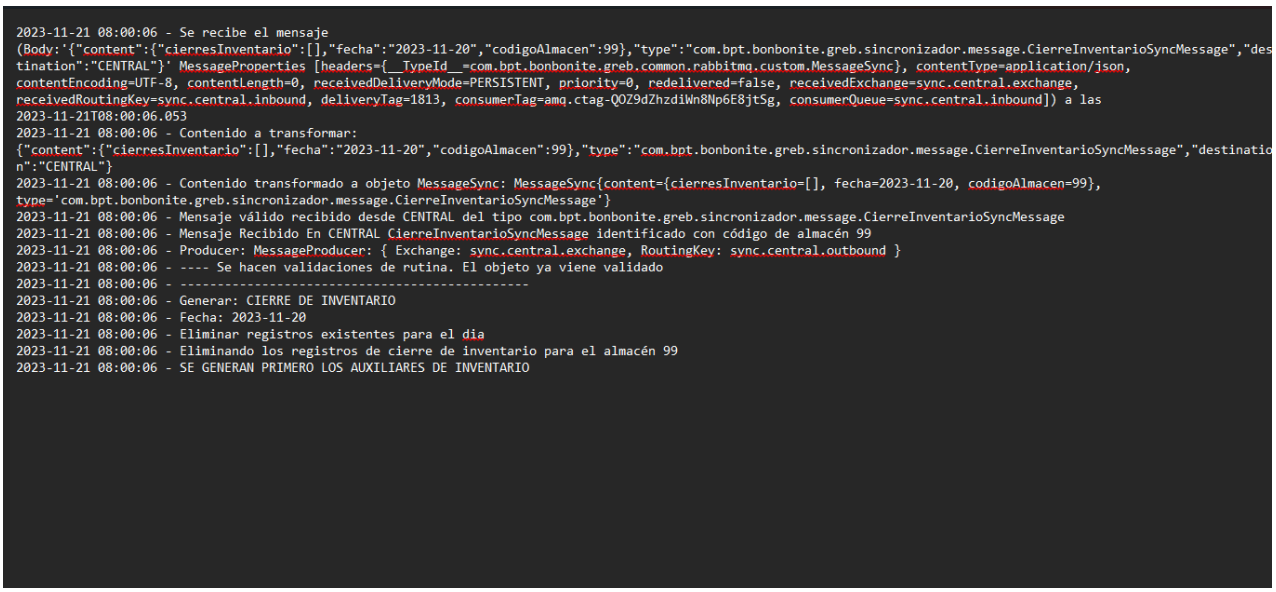

Imagen número 14 (Log de prueba)

Luego de haber instalado y configurado las características de Logstash se procede a configurar un índice con las mismas características que se configuraron en el archivo ".conf" de Logstash, esto con el fin de que Elastic pueda diferenciar los tipos de datos obtenidos.

Nota: Gracias a la versatilidad de las tecnologías ELK se pueden configurar varios tipos de índices según la necesidad del usuario, por lo que no se estaría limitado a un solo tipo de archivo y no solo los logs podrían ser leídos sino cualquier tipo de archivo que se requiera.

Para la configuración de dicho índice lo que se hizo fue consumir un servicio enviando un índice de prueba con el tipo de información que se espera recibir del archivo log, lo que se puede visualizar en la imagen número 14 a continuación

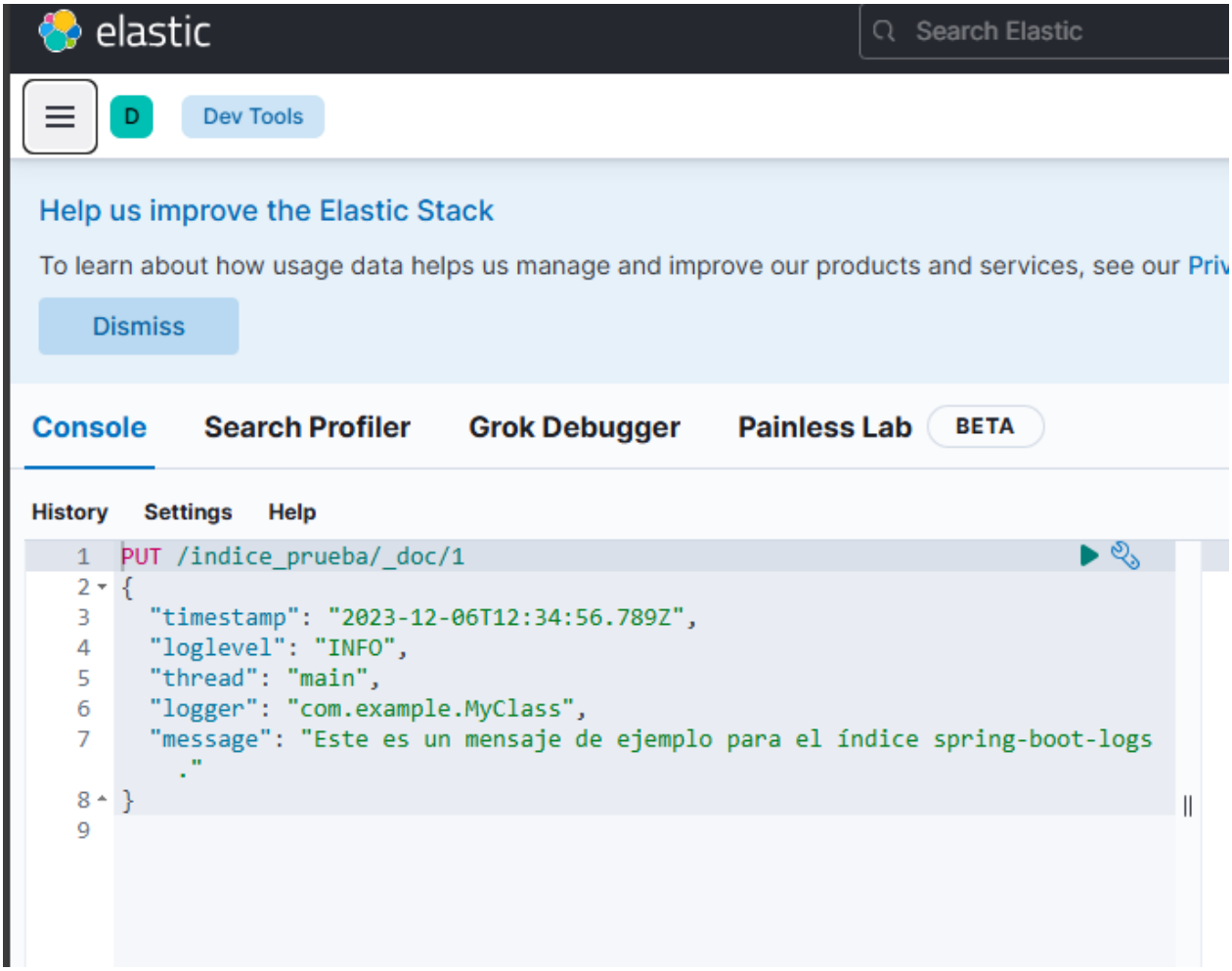

Imagen número 15 (Creación de índice de logs)

Luego de esto se creó un patrón de índices para que se pueda leer y visualizar correctamente la información recopilada por Logstash y Elastic

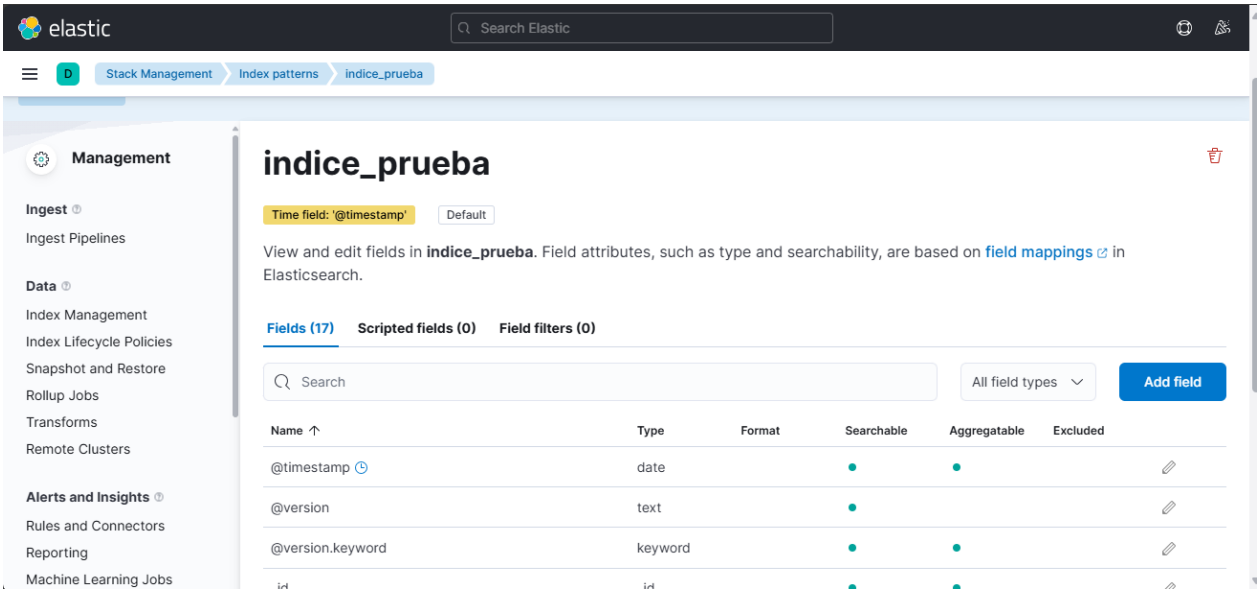

Imagen número 16 (Patrón de índices)

Con todo este proceso configurado se obtiene cada cambio realizado en el archivo .log configurado en la sección Discover de Kibana.

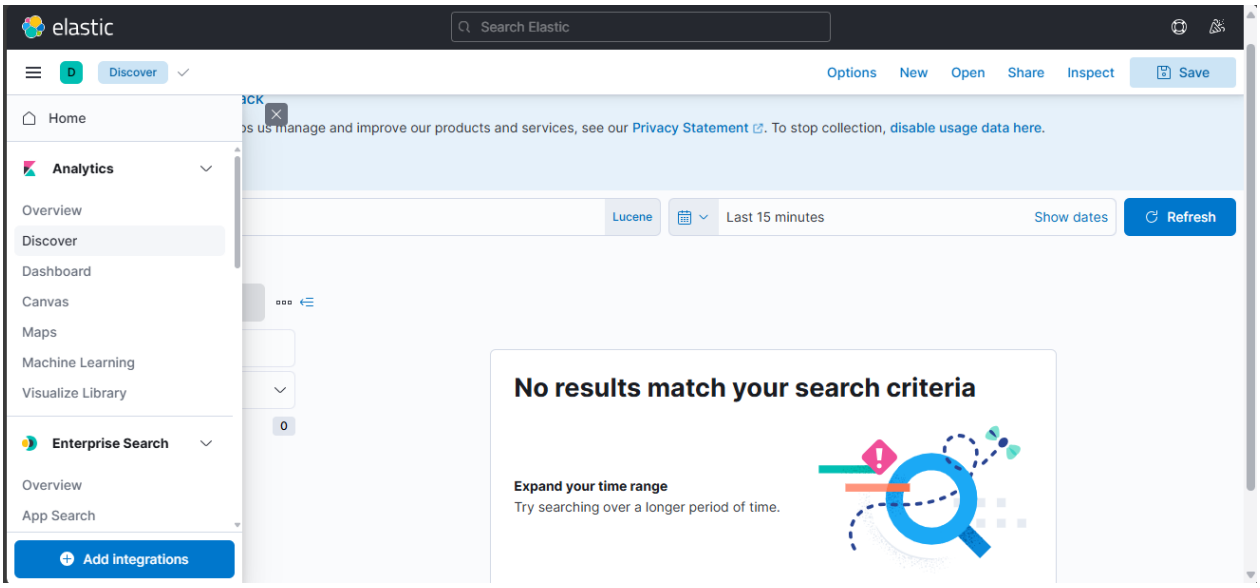

Imagen número 17 (Información filtrada)

Luego de esto se procede a realizar un cambio en el archivo .log y verificar si posteriormente se subió y se puede visualizar en Kibana. Se añadieron datos con la fecha y la hora actual y con un mensaje de prueba para que su filtro y diferenciación sea posible.

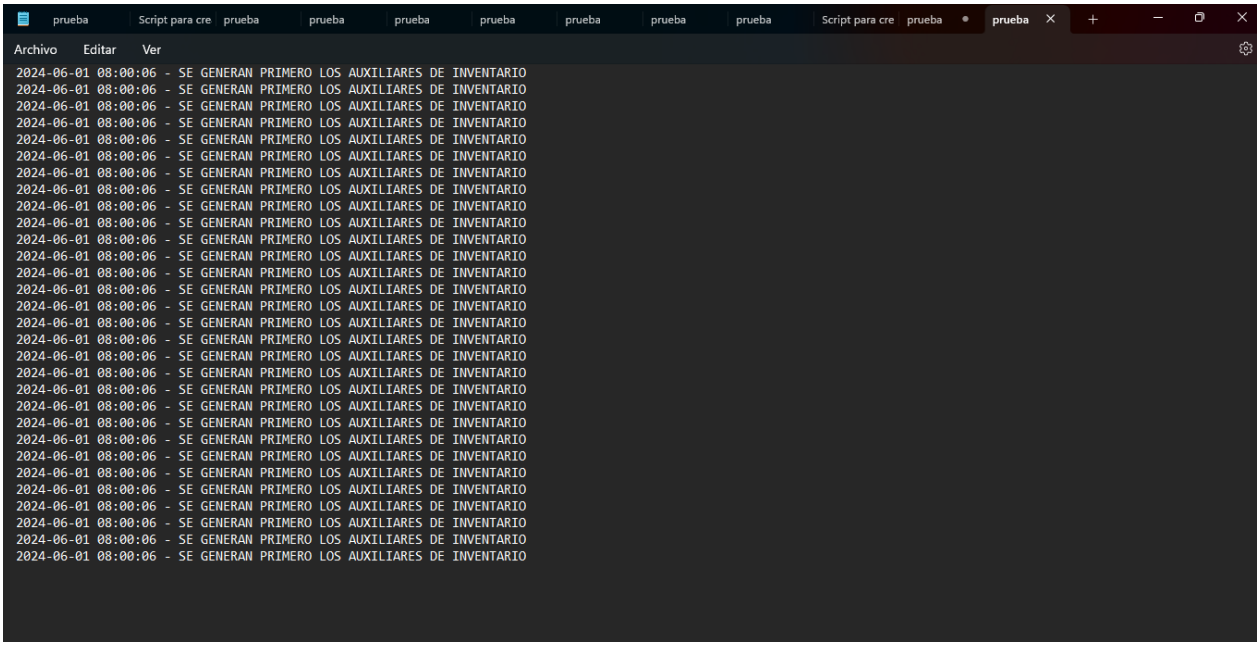

Imagen número 18 (log de prueba con información adicional)

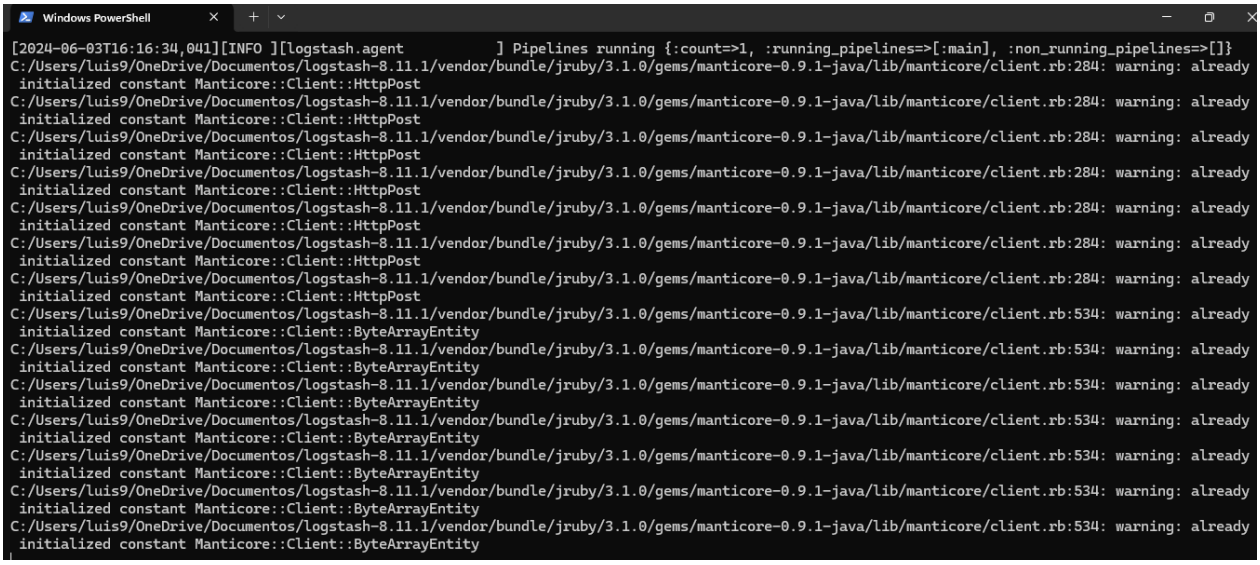

Imagen número 19 (log de prueba con información adicional)

En la imagen número 18 se puede ver como Logstash al detectar los cambios realizados dentro del archivo log configurado inmediatamente los envía por medio de servicios REST haciendo un post

al endpoint de Elasticsearch, si todo salió exitosamente esto se podrá visualizar y filtrar en el overview de Kibana.

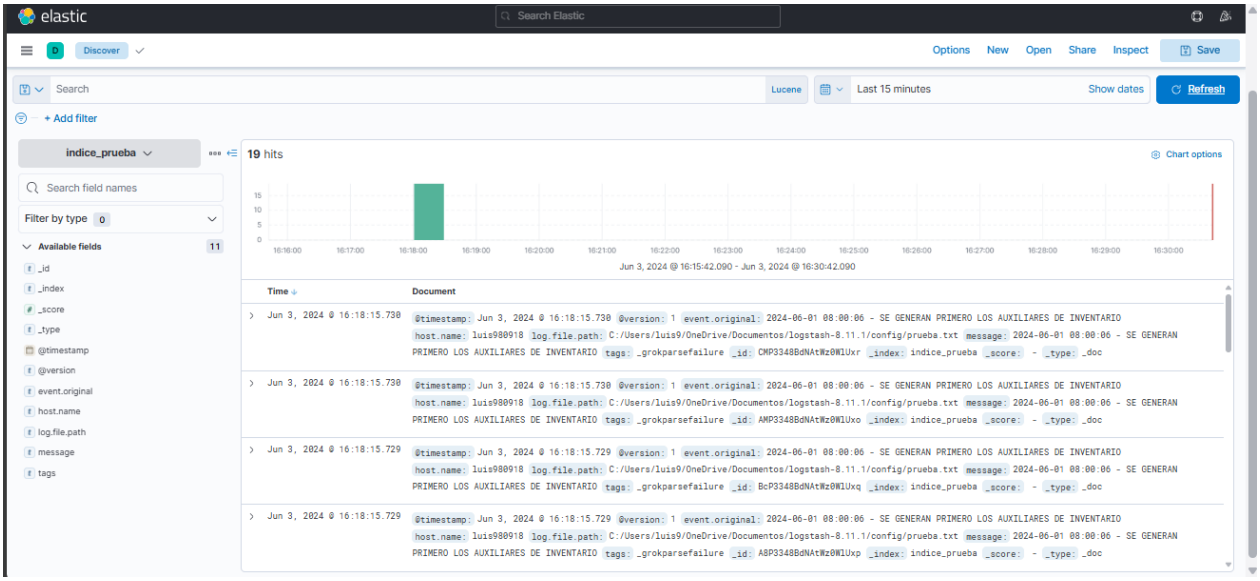

Imagen número 20 (Información filtrada y recopilada)

En la imagen número 19 se puede observar entonces cómo se subió la información y se almacenó dentro del índice creado, luego de esto se pudo filtrar por la información subida durante los últimos 15 minutos, así mismo se puede configurar cualquier tipo de filtro sea por fecha, hora, tipo de mensaje (WARNING, ERROR, INFORMATION, etc…), lo ideal sería que cuando todo esto se haga efectivo en el entorno productivo se cree un índice por cada almacén o por cada instancia del aplicativo para tener toda la información lo más organizada posible y para que aun estando toda la información centralizada pueda estar también en un orden bien especificado y que sus filtros se hagan de la forma más usable posible.

Luego de tener toda la información centralizada el paso siguiente fue establecer métricas que ayuden a visualizar de una mejor manera la información recopilada, para este caso se configuró una en donde se establece en un rango de tiempo determinado el número de registros ingresados, el caso que se mostrará a continuación podrá reflejar la información ingresada todo este año y como para el caso de pruebas originalmente se subían pequeños fragmentos de logs en la gráfica se ve un patrón muy bajo al principio y muy alto al final y esto es debido a que se ingresó la información real de uno de los almacenes con más de 1000 registros y se visualiza en un patrón de ascenso bastante elevado.

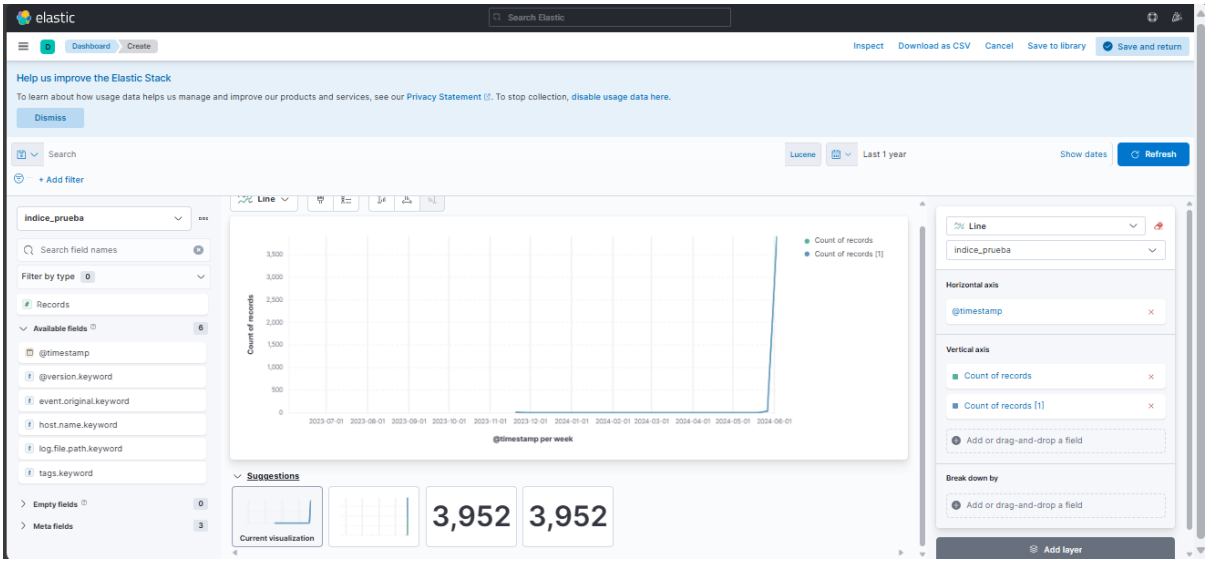

Imagen número 21 (Configuración de métricas)

Así mismo se generó un histograma que permite visualizar detalladamente todas las fechas en las que se llegaron a subir registros, mostrándolos de manera organizada y siendo de gran ayuda en caso de filtrar la información por fecha, la verdadera importancia se puede ver reflejada en casos en los que llegan a reportar algún inconveniente de algún almacén a cierta hora del día, filtrar esto gracias a ELK sería demasiado fácil, ya que se podría generar una visualización como la de la imagen 21 filtrando por las últimas horas y seleccionando la data justo a la hora en la que se realizó el reporte, permitiendo visualizar el log de error para un posterior análisis y solución.

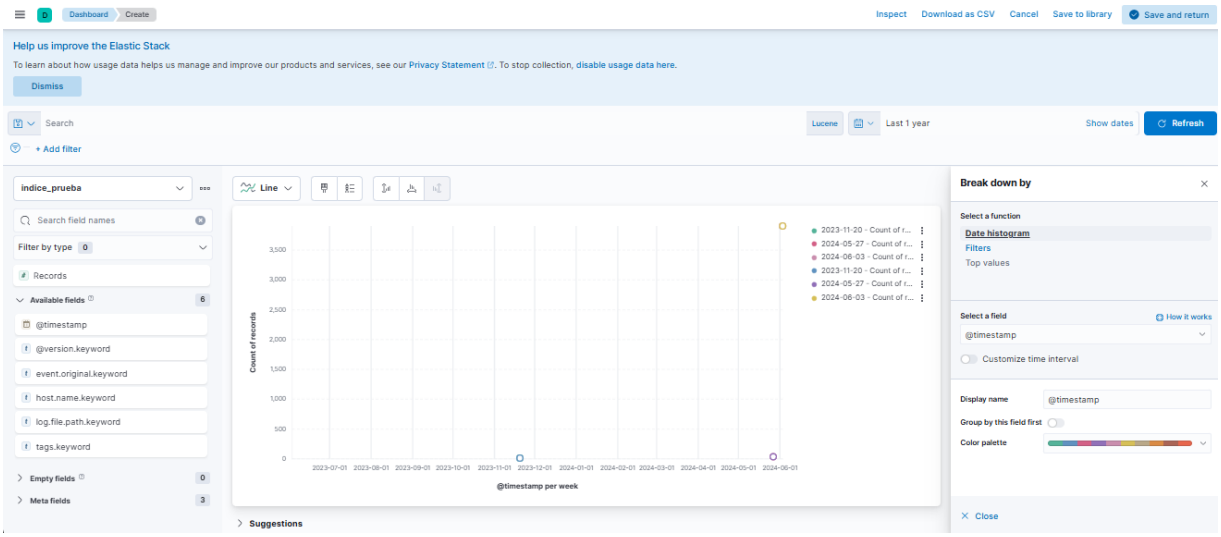

Imagen número 22 (Histograma con información sobre fechas donde se ingresaron registros)

7. DISCUSIÓN

Sobre la efectividad de las estrategias que se han puesto en práctica, las estrategias de observabilidad y monitoreo continuo son de gran utilidad para mejorar la proactividad en la detección y respuesta a problemas operativos. Se han utilizado herramientas como ELK para detectar varios problemas y tomar medidas con prontitud, aunque como se especificó en las secciones anteriores del informe solo se hicieron efectivas en entornos locales, esto no significa que no hayan sido útiles ya que en estos tipos de procesos de ejecución también se puede llegar a tener la ocurrencia de errores que fácilmente pueden ser filtrados y leídos gracias a las tecnologías utilizadas, pero aclarando que no es tanto la tecnología utilizada sino la arquitectura de observabilidad por medio de la cual se basó la implementación e instalación de la tecnología que no fue más que el medio para lograr los objetivos planteados al principio.

Durante el proceso de implementación ocurrieron muchos inconvenientes y desafíos que fueron muy significativos, como el aprendizaje y el uso de tecnologías nunca utilizadas en la universidad lo que implicaba un conocimiento muy bajo sobre el uso e implementación de ellas o también la falta de documentación sobre la configuración de las mismas, pero esto no fue un stopper significativo, sino por el contrario sirvió de ayuda para aprender nuevas cosas además que proporcionaron valiosas lecciones de aprendizaje.

En el futuro se piensa llevar a cabo esta implementación al proceso en productivo de la aplicación y masificar todos los conocimientos obtenidos, además de generar nuevos tipos de métricas que ayuden a identificar bugs y patrones que faciliten la solución de los mismos e incluso prevenirlos con el fin de tener una mejora continua en los procesos nuevos y existentes.

8. CONCLUSIONES

Gracias a las investigaciones realizadas sobre los flujos de mantenimiento se pudo obtener una base sólida sobre la cual fundamentar todas y cada una de las estrategias de observabilidad planteadas, además de servir de guía para su análisis posterior y su implementación, por ende, se concluye que los procesos de mantenimiento deberían ir integrados estrechamente con las estrategias de supervisión continua para garantizar el correcto funcionamiento de cualquier tipo de producto de software.

Es bastante importante dejar por sentado que el éxito de la implementación de la solución de observabilidad no se limita únicamente en la tecnología utilizada, sino que depende en gran medida del análisis y estudio detallado de la arquitectura de la implementación propuesta, todos y cada uno de los objetivos planteados al inicio de este informe sirvieron como guía para establecer una estrategia muy robusta que ayudó a abordar los aspectos más importantes del fin esperado, si bien ELK fue una magnífica herramienta para llevar a cabo el exitoso desarrollo de esta práctica el verdadero valor radica en cómo se integra dentro de una arquitectura coherente y adaptada a las necesidades específicas de una correcta supervisión.

Al inicio de este proyecto se pudo destacar la capacidad de trabajo en equipo como un pilar fundamental para la implementación exitosa de la tecnología de supervisión continua, en este caso ELK ya que se tuvo gran colaboración de compañeros de trabajo con más experiencia en el área que fueron indispensables para dar un primer paso, además de contar también con el apoyo del equipo de operaciones que permitió un excelente análisis de requisitos y planificación para dar con la mejor arquitectura para la solución.

Sin embargo, a medida que avanzaba el proyecto, hubo un cambio significativo en la dinámica del equipo, lo que implicó un desafío individual tanto así que se convirtió en una labor de liderazgo para dar con gran parte de la solución. Aunque esta transición presentó sus propios desafíos, también fue clave para destacar habilidades como la resolución de problemas y la toma de decisiones, además de resaltar también la gran importancia de la adaptabilidad y la flexibilidad en entornos dinámicos de proyectos, el compromiso con los objetivos del proyecto se mantuvo constante, lo que permitió superar todos los desafíos que surgieron y obtener así resultados exitosos.

9. REFERENCIAS

[1] TRACTIAN. (s.f.). Patrón de Fallo: la importancia en el mantenimiento. Recuperado de https://blog.tractian.com/es/patron-de-fallo-la-importancia-en-el-mantenimiento

[2] Zavalar, J. (2024). AYDS: A New Approach to Weight Loss. Angelfire. Recuperado el 17 de marzo de 2024, de https://www.angelfire.com/scifi/jzavalar/far/ayds.html

[3] González, M. (2024). El Papel de las Pruebas Automatizadas en la Detección Temprana de Bugs. Blog de Marta González. Recuperado el 14 de marzo de 2024, de https://martagonzalez.dev/blog/el-papel-de-las-pruebas-automatizadas-en-ladeteccion-temprana-de-bugs/

[4] guywi-ms, “Introducción a Log Analytics en Azure Monitor - Azure Monitor | Microsoft Learn.” [Online]. Available: https://learn.microsoft.com/es-es/azure/azure-monitor/logs/log-analytics-overview

[5] N. author found, “Get started with Grafana and Prometheus | Grafana documentation - Grafana Labs.” [Online]. Available: https://grafana.com/docs/grafana/latest/getting-started/get-started-grafana-prometheus/

[6] Grafana, “What is Prometheus? | Grafana documentation - Grafana Labs.” [Online]. Available: https://grafana.com/docs/grafana/latest/fundamentals/intro-to-prometheus/

[7] Grafana, “Attention Required! | Cloudflare.” [Online]. Available: https://medium.com/@ismaelaguilera_/instalaci%C3%B3n-y-configuraci%C3%B3n-de-prometheus-grafana-centos8-331c0e43ccc1

[8] Elastic. (s.f.). Casos de uso: Historias de éxito del Elastic Stack. Recuperado el 17 de marzo de 2024, de https://www.elastic.co/customers

[9] halkazwini, “Análisis de registros de flujo de grupo de seguridad de red de Azure con Graylog | Microsoft Learn.” [Online]. Available: https://learn.microsoft.com/es-es/azure/network-watcher/network-watcher-analyze-nsg-flow-logs-graylog

[10] Splunk, “Splunk | The Key to Enterprise Resilience.” [Online]. Available: https://www.splunk.com/

[11] Followers, “Componentes Básicos de una Arquitetura SPLUNK.” [Online]. Available: https://www.linkedin.com/pulse/componentes-b%C3%A1sicos-de-una-arquitetura-splunk-palacios-olivar/

[12] Gonzalez, Santiago, “Attention Required! | Cloudflare.” [Online]. Available: https://xantygc.medium.com/por-qu%C3%A9-escribir-logs-de-calidad-3dd6e721569d

[13] Walberth, “Buenas Practicas Logging - Coding With No Try Catch - Coding With No Try Catch - When the program doesn’t need try catch.”. [Online]. Available: https://codingwithnotrycatch.com/2019/10/21/buenas-practicas-logging/

[14] Elasticsearch, “Elasticsearch: Motor de búsqueda y analítica distribuido oficial | Elastic.” [Online]. Available: https://www.elastic.co/es/elasticsearch