



**MRS**

# **MONITOREO Y RENDIMIENTO DE SOFTWARE**

**INGENIERIA DE SISTEMAS  
COMPUTACIONALES**

## **INTEGRANTES:**

**JOSE ANTONIO SANCHEZ CERDA**

**ALEXIS DAVID CEBALLOS CADENA**

**ERICK JOVANNY RAMIREZ AGUILAR**

**ERICK SALVADOR VELOZ gonzalez**



---

## RESUMEN EJECUTIVO

---

Nuestro proyecto llamado MRS, es un programa que nos servirá para automatizar las consultas del rendimiento del CPU, generando un script en Excel, además de generar las tablas con las métricas, con ayuda de la inteligencia artificial Copilot, el usuario hace preguntas acerca del documento y la IA ayudara a dar respuesta y fundamentar lo ya cuestionado anteriormente, a través de la medición del rendimiento evaluaremos la velocidad, eficiencia y estabilidad de nuestro software, al detectar anomalías implementaremos sistemas de alerta para identificar fallos en tiempo real, mediante la optimización continua se utilizaran las métricas para mejorar el rendimiento del programa. Además de Copilot utilizaremos herramientas como Prometheus, este recopilara las métricas utilizadas y las alertas personalizadas; al igual que Elasticsearch, este sistema nos ayudara a compensar las pérdidas de valor a largo plazo generadas por la devaluación, así como también para generar la búsqueda de registros; el programa Logstash, automatiza y transforma los datos; Kibana, ayuda a la exploración y el análisis de los registros. Durante el desarrollo del proyecto se identificarán los cuellos de botella para eficientar el tiempo de ejecución y optimizar los recursos; para mantener la fiabilidad del proyecto, se detectarán los problemas antes de que los usuarios resulten afectados. Nuestro enfoque en el monitoreo y rendimiento de software nos permitirá ofrecer una experiencia excepcional a nuestros usuarios y mantener la calidad de nuestros sistemas. El costo de la aplicación es de \$350 MXN. mensuales más \$160 MXN por servicios de Microsoft, MRS ofrece una solución completa y accesible para empresas de distintos tamaños.

---

*CARTA PRESENTACION*

---

MRS

Ecatepec, Edo. Mex. Agosto, 2024

Estimado equipo y partes interesadas, dirijo a ustedes la presente carta, con el fin de presentar nuestro proyecto de “Monitoreo y rendimiento de Software”, enfocado al área de sistemas.

Nuestros objetivos están enfocados en optimizar la eficiencia y potencializar la confianza en las aplicaciones y en la creación de nuevos sistemas.

Nuestro proyecto se especializa en automatizar consultas en el rendimiento del CPU, con ayuda de las herramientas de inteligencia artificial, que nos proporcionaran una visión más amplia de nuestro entorno, nos alertaran ante posibles problemas y nos ayudara a optimizar el rendimiento de nuestro software.

Exponiendo ideas competentes y estando al alcance de todos; MRS aspira a ser una empresa de nivel nacional y posicionarnos en el mercado mexicano, con el paso del tiempo seguiremos innovando a través del rendimiento del software para facilitar a todo usuario que desee utilizar nuestro programa, evitando problemas del sistema y haciendo consultas de manera eficiente.

Esta carta va dirigida a toda aquella empresa que requiera tener un MRS y hacer consultas de forma automatizada con la ayuda de la inteligencia artificial de forma segura y rápida.

¡Estamos entusiasmados por el éxito de nuestro proyecto y esperamos colaborar con todos ustedes!

Agradecemos su atención.

Atentamente:

Antonio Sánchez Cerda

Alexis David Ceballos Cadena

Erick Jovanny Ramírez Aguilar

Erick Salvador Veloz González

# INDICE

TABLA DE ILUSTRACIONES.....	6
MARCO CONTEXTUAL .....	7
Entorno Económico y Tecnológico .....	7
Protocolo.....	8
Configuración de Herramientas de Monitoreo: .....	9
Integración de Copilot 365:.....	9
Despliegue y Escalabilidad: .....	9
Análisis y Mejoras Continuas: .....	9
Evaluación del Impacto y Retroalimentación:.....	9
Justificación .....	10
Contexto Económico y Tecnológico.....	10
Uso de Computadoras y Dispositivos Móviles: .....	10
Penetración de TIC en las Empresas .....	10
Necesidad del Proyecto.....	10
Problemática .....	11
Detección Temprana de Problemas de Rendimiento:.....	11
Complejidad en la Gestión de Logs y Métricas: .....	11
Desafíos en la Configuración y Mantenimiento de Herramientas de Monitoreo:.....	11
Identificar y resolver la causa de la lentitud en aplicaciones:.....	11
Crecimiento del Comercio Electrónico:.....	12
Objetivos .....	12
1.    Optimizar el Rendimiento: .....	12
2.    Monitoreo en Tiempo Real: .....	12
3.    Análisis de Datos .....	12
4.    Visualización Clara:.....	12
5.    Mejora Continua: .....	12
Variable de Innovación .....	12
Innovación en Herramientas de Monitoreo.....	12
Impacto de la Innovación .....	12
MARCO TEORICO.....	14
Monitoreo de Contenedores: .....	15
1.2. Monitoreo de Software .....	15
1.3. Rendimiento de Software .....	15
1.4. Integración de IA en Microsoft 365.....	16
1.5. Impacto de Copilot en el Monitoreo y Rendimiento .....	16
1.6. Mantenimiento de Software .....	16
Rendimiento de Software .....	17
Integración de Copilot con Graph y Aplicaciones Microsoft 365 .....	18

¿Cómo funciona Microsoft Copilot para Microsoft 365? .....	18
Confiabilidad y seguridad.....	21
Tecnologías a implementar .....	23
Copilot 365.....	23
Uso en el Proyecto: .....	23
Prometheus.....	23
Uso en el Proyecto: .....	23
Elastic Stack (ELK).....	24
Uso en el Proyecto: .....	24
WMI .....	25
Arquitectura de WMI .....	25
Prometheus.....	26
PHP .....	27
Herramientas de Alertas .....	28
PagerDuty.....	28
Opsgenie .....	28
Uso en el Proyecto: .....	28
Lenguajes de Programación y Scripting .....	28
Python.....	28
Bash.....	28
JavaScript .....	28
Uso en el Proyecto: .....	29
Implementación del Proyecto .....	29
Prometheus.....	29
Integración y Despliegue.....	29
(service, 2023) <a href="https://www.soluciones-im.com/es/la-monitorizacion-de-sistemas-informaticos-en-8-pasos">https://www.soluciones-im.com/es/la-monitorizacion-de-sistemas-informaticos-en-8-pasos</a> .....	29
Uso del Usuario .....	35
Beneficios Actuales del Monitoreo y Rendimiento de Software .....	43
Metodología .....	44
¿Qué es la metodología Prototipó? .....	44
Etapas de la metodología Prototipo .....	44
DIAGRAMA DE FLUJO DE DATOS .....	47
DIAGRAMA DE CASO DE USO .....	48
VISUALIZACIÓN DE RENDIMIENTO .....	48
Diagrama de Flujo Lógico .....	49
Diccionario de Datos .....	50
ESTIMACIÓN ECONÓMICA PROYECTO .....	51
Código .....	52
Alternativas .....	62
Trazas distribuidas.....	62
Alertas: .....	62

Análisis de errores:.....	62
Dashboards personalizables.....	62
Monitoreo en tiempo real.....	62
Alertas configurables .....	62
Dashboards personalizables.....	63
Integración con múltiples servicios.....	63
Monitoreo de usuario final .....	63
Análisis de rendimiento de aplicaciones.....	63
Diagnóstico de problemas.....	63
Integración con herramientas DevOps.....	63
Componentes Clave de nuestra Plataforma.....	63
Fluentd: .....	64
Manual de Usuario o Guion de pruebas .....	65
Monitoreo de Rendimiento de Software (MRS) .....	65
Pasos para usar el programa MRS.....	65
Consultar CPU: .....	65
Retos y Esfuerzos.....	66
Conclusiones .....	68
BIBLIOGRAFIA.....	70
GRACIAS .....	71

## TABLA DE ILUSTRACIONES

Ilustración 1 copilot.....	20
Ilustración 2 Arquitectura de WMI.....	25
Ilustración 3 Arquitectura de Prometheus.....	27
Ilustración 4 DIAGRAMA DE DATOS .....	47
Ilustración 5 Diagrama de Flujo de Datos .....	49
Ilustración 6 Diccionario de Datos .....	50
Ilustración 7 código 1 .....	52
Ilustración 8 código 2 .....	53
Ilustración 9 codigo 3 .....	53
Ilustración 10 código 4 .....	53
Ilustración 11 código 5 .....	54
Ilustración 12 codigo 6 .....	54
Ilustración 13 codigo 7 .....	54
Ilustración 14 codigo 8 .....	55
Ilustración 15 codigo 9 .....	56
Ilustración 16 codigo 10 .....	57
Ilustración 17 codigo 11 .....	57
Ilustración 18 codigo 12 .....	57
Ilustración 19 codigo 13 .....	58
Ilustración 20 codigo 14 .....	58
Ilustración 21 codigo 15 .....	59
Ilustración 22 codigo 16 .....	60
Ilustración 23 codigo 17 .....	60

## MARCO CONTEXTUAL

El monitoreo y rendimiento de software son esenciales para el desarrollo y sostenibilidad de las empresas en la economía digital actual. La implementación de herramientas avanzadas de monitoreo no solo mejora la eficiencia operativa y la experiencia del usuario, sino que también asegura el cumplimiento regulatorio y la seguridad. Este marco teórico proporciona una guía para la inversión y justificación del proyecto, destacando la relevancia y el impacto positivo que puede tener en las empresas y en la economía en general.

En la economía digital actual, la velocidad lo es todo, especialmente cuando hay lentitud en aplicaciones y sitios web a los que acceden miles de usuarios. Esto tiene un impacto directo y perjudicial en la productividad del negocio, las ganancias e incluso en la propia imagen de la marca.

Dada la complejidad de las infraestructuras y las aplicaciones actuales, esto no es fácil de lograr. Las plataformas de desarrollo de aplicaciones móviles, las infraestructuras de Nube, los servidores virtualizados y en contenedores, las arquitecturas de aplicaciones IoT y un largo etcétera hacen que la gestión del rendimiento sea un desafío.

Además, con las diversas interdependencias entre componentes, es difícil identificar la causa de la lentitud de la aplicación. Una de las preguntas más difíciles que enfrentan los propietarios de aplicaciones, desarrolladores y gerentes de TI es "¿por qué mi aplicación es lenta?" para responder a esta pregunta debemos considerar múltiples factores como si la infraestructura de nube que usamos está sufriendo una degradación o si existe algún otro factor externo que esté afectando el rendimiento o es propiamente un problema interno.

### Entorno Económico y Tecnológico

México es la segunda economía más grande de América Latina y ha visto un crecimiento significativo en su sector tecnológico. Con un mercado creciente en comercio electrónico, fintech, telecomunicaciones y servicios digitales, la infraestructura tecnológica se ha convertido en un pilar esencial para el desarrollo económico del país.



## Protocolo

Identificar los conceptos fundamentales de monitoreo de rendimiento de software y la funcionalidad de Copilot, Reconocer y establecer las métricas clave que se utilizan en el monitoreo de rendimiento de software en base a las necesidades de la empresa.

Explicar cómo Copilot puede integrarse en una plataforma de monitoreo de rendimiento de software.

Describir los beneficios de utilizar Copilot para el análisis de rendimiento de software.

Implementar una forma básica de Copilot en la plataforma de monitoreo.

Utilizar Copilot para obtener datos de rendimiento de software y generar alertas en caso de anomalías presentando cada uno de los casos en un formato comprensible para los desarrolladores.

Analizar los datos de rendimiento recopilados por Copilot para identificar patrones y áreas de mejora o para erradicar posibles anomalías en el sistema.

Diseñar y desarrollar funcionalidades adicionales para la plataforma que utilicen los datos proporcionados por Copilot para generar informes detallados y personalizados en base al gusto o necesidad de los desarrolladores.

Desarrollar una interfaz intuitiva que a su vez genere recomendaciones basadas en los análisis de rendimiento realizados por Copilot.

Evaluar el impacto de la plataforma en el proceso de desarrollo de software mediante estudios de caso y retroalimentación de los usuarios y desarrolladores.

Valorar la precisión y utilidad de los informes generados por la plataforma y proponer mejoras basadas o soluciones en caso de anomalías en los resultados obtenidos.

El proyecto de monitoreo de rendimiento de software tiene como objetivo implementar un sistema integral de monitoreo y análisis del rendimiento de aplicaciones y sistemas informáticos. Este sistema se basará en un conjunto de herramientas y tecnologías específicas, incluyendo Prometheus, Elasticsearch, Logstash, Kibana, Copilot 365, entre otras. El alcance del proyecto incluye:



#### Configuración de Herramientas de Monitoreo:

Implementación y configuración de Prometheus para la recolección de métricas de rendimiento, incluyendo tiempo de respuesta, rendimiento del sistema, tasa de errores, entre otros.

Configuración de copilot para la visualización de métricas en dashboards personalizados, proporcionando una representación visual clara del estado del sistema. Implementación de Elasticsearch, Logstash y Kibana para la recolección, almacenamiento y visualización de logs de aplicaciones y sistemas.

#### Integración de Copilot 365:

Integración de Copilot 365 en el entorno de desarrollo de Visual Studio y Visual Studio Code para asistir en la escritura de scripts de monitoreo en Python y Bash.

Utilización de Copilot 365 para la configuración de dashboards y alertas en Prometheus, facilitando la automatización de tareas rutinarias.

#### Despliegue y Escalabilidad:

Implementación del sistema de monitoreo en un entorno Dockerizado utilizando Docker y Docker Compose para simplificar la gestión y escalabilidad de los componentes. Configuración de los diferentes servicios (Prometheus, Elasticsearch, Logstash) en archivos ``docker-compose.yml`` para facilitar su despliegue y administración.

#### Análisis y Mejoras Continuas:

Análisis de los datos de rendimiento recopilados por las herramientas de monitoreo para identificar patrones, áreas de mejora y posibles anomalías en el sistema.

Implementación de funcionalidades adicionales en la plataforma para generar informes detallados y personalizados en base a las necesidades de los desarrolladores.

#### Evaluación del Impacto y Retroalimentación:

Evaluación del impacto de la plataforma en el proceso de desarrollo de software mediante estudios de caso y retroalimentación de usuarios y desarrolladores. Valoración de la precisión y utilidad de los informes generados por la plataforma, proponiendo mejoras basadas en la retroalimentación y análisis de los resultados obtenidos.

# Justificación

## Contexto Económico y Tecnológico

México, siendo la segunda economía más grande de América Latina, ha experimentado un crecimiento significativo en su sector tecnológico. La digitalización y la adopción de tecnologías avanzadas han aumentado la complejidad de los sistemas de TI, lo que hace que el monitoreo efectivo sea crucial para asegurar el rendimiento y la seguridad.

Uso de Computadoras y Dispositivos Móviles: La misma encuesta reveló que el 78.3% de la población utilizaba teléfonos inteligentes, y el 45.3% computadoras. Estos datos son relevantes para entender las plataformas y dispositivos que predominan en el mercado, impactando las estrategias de desarrollo y monitoreo de software.

Penetración de TIC en las Empresas: Según el Censo Económico 2019, el 72.9% de las unidades económicas usaban computadoras y el 57.5% tenían acceso a internet. Estos datos destacan la adopción de tecnologías digitales en el entorno empresarial, reforzando la relevancia del monitoreo y gestión del rendimiento de software en las operaciones empresariales.

## Necesidad del Proyecto

La velocidad y eficiencia de las aplicaciones y sitios web son esenciales en la economía digital actual. La lentitud en el rendimiento afecta negativamente la productividad, las ganancias y la imagen de marca. Dada la complejidad de las infraestructuras modernas, es fundamental contar con sistemas avanzados de monitoreo que permitan identificar y resolver problemas de rendimiento de manera efectiva.

(INEGI, 2017 a 2020 )

## Problemática

La creciente complejidad de las aplicaciones y sistemas informáticos ha generado una demanda creciente de herramientas eficientes de monitoreo y análisis de rendimiento. Las organizaciones enfrentan desafíos para garantizar que sus aplicaciones funcionen de manera óptima, manteniendo altos estándares de rendimiento y disponibilidad. Algunos de los principales problemas que este proyecto busca abordar incluyen:

### Detección Temprana de Problemas de Rendimiento:

La falta de visibilidad sobre el rendimiento de las aplicaciones puede resultar en la detección tardía de problemas, lo que afecta negativamente la experiencia del usuario y la eficiencia operativa.

### Complejidad en la Gestión de Logs y Métricas:

La diversidad de sistemas y aplicaciones utilizados en entornos empresariales genera grandes volúmenes de logs y métricas que pueden ser difíciles de gestionar y analizar de manera eficiente.

### Desafíos en la Configuración y Mantenimiento de Herramientas de Monitoreo:

La configuración manual y el mantenimiento de herramientas de monitoreo pueden ser laboriosos y propensos a errores, especialmente en entornos de infraestructura escalables y dinámicos.

### Dificultades en la Identificación de Causas Raíz de Problemas de Rendimiento:

La falta de herramientas efectivas para el análisis de datos de rendimiento puede dificultar la identificación rápida y precisa de las causas raíz de los problemas de rendimiento.

### Necesidad de Mejoras Continuas y Optimización de Recursos:

La optimización del rendimiento de las aplicaciones es un proceso continuo que requiere la identificación constante de áreas de mejora y la implementación de soluciones eficientes.

Identificar y resolver la causa de la lentitud en aplicaciones: es un desafío crítico para propietarios de aplicaciones, desarrolladores y gerentes de TI. La degradación de la infraestructura de nube, factores externos y problemas internos complican esta tarea.

Los enfoques tradicionales de supervisión no son suficientes en los entornos híbridos actuales. Las organizaciones deben conectar el viaje del usuario con la infraestructura de la aplicación para comprender cuándo, dónde y por qué la experiencia del usuario se ve afectada.

**Crecimiento del Comercio Electrónico:** En 2019, el comercio electrónico en México creció un 28.6%, según datos del INEGI. Este crecimiento enfatiza la importancia de asegurar que las plataformas de comercio electrónico funcionen de manera eficiente y segura, subrayando la necesidad de monitoreo continuo.

## Objetivos

1. **Optimizar el Rendimiento:** Mejorar la eficiencia operativa del software identificando y resolviendo cuellos de botella.
2. **Monitoreo en Tiempo Real:** Establecer un sistema de monitoreo en tiempo real para la detección temprana de problemas.
3. **Análisis de Datos:** Implementar soluciones avanzadas de análisis de logs y métricas.
4. **Visualización Clara:** Proporcionar visualizaciones detalladas del rendimiento del software para facilitar el análisis y la toma de decisiones.
5. **Mejora Continua:** Asegurar la mejora continua del sistema a través de evaluaciones periódicas y ajustes necesarios.

## Variable de Innovación

### Innovación en Herramientas de Monitoreo

La implementación de herramientas avanzadas como Copilot 365, Prometheus, Elasticsearch, Logstash y Kibana representa una innovación significativa en el monitoreo del rendimiento de software. Estas herramientas permiten un monitoreo en tiempo real, análisis detallados y visualizaciones intuitivas, superando las limitaciones de los enfoques tradicionales.

### Impacto de la Innovación

1. **Eficiencia Operativa:** Mejora la eficiencia operativa mediante la rápida identificación y resolución de problemas.

2. Seguridad y Cumplimiento: Ayuda a cumplir con regulaciones estrictas y mejorar la seguridad informática.
3. Satisfacción del Usuario: Mejora la experiencia del usuario al asegurar un rendimiento óptimo de las aplicaciones.
4. Competitividad: Permite a las empresas mexicanas competir a nivel global ofreciendo servicios de alta calidad.
5. Crecimiento Económico: Contribuye al crecimiento económico al permitir a las empresas operar con mayor eficiencia y reducir costos operativos.



# MARCO TEORICO

El monitoreo de sistemas y aplicaciones ha experimentado una evolución significativa a lo largo del tiempo, adaptándose continuamente a las crecientes demandas de las infraestructuras tecnológicas modernas. Desde sus primeros días junto con los mainframes en las décadas de 1960 y 1970, donde el foco estaba en el rendimiento del hardware como la memoria y la CPU, hasta la era actual de la computación en la nube y los contenedores, las herramientas de monitoreo han evolucionado en respuesta a los avances tecnológicos y las necesidades empresariales emergentes.

Con la expansión de las redes informáticas en las décadas de 1980 y 1990, surgió la necesidad de monitorear sistemas distribuidos y redes más complejas. Esto llevó al desarrollo de herramientas más sofisticadas que podían supervisar múltiples nodos y recursos de red de manera simultánea, mejorando la capacidad de los administradores de IT para detectar y resolver problemas de manera más eficiente.

En la década de 2000, con la popularización de Internet y la adopción masiva de la computación en la nube, las herramientas de monitoreo se enfrentaron al desafío de gestionar entornos dinámicos y escalables. Surgieron tecnologías de código abierto como Nagios, Zabbix y Munin, que ofrecían flexibilidad y personalización, permitiendo a las empresas adaptar sus soluciones de monitoreo a sus necesidades específicas sin depender exclusivamente de proveedores comerciales.

Avanzando hacia el presente, el monitoreo en la nube y de contenedores ha redefinido las expectativas de las herramientas de monitoreo. Plataformas como Docker y Kubernetes han transformado la manera en que las aplicaciones son implementadas y gestionadas, requiriendo herramientas de monitoreo que puedan ofrecer visibilidad granular dentro de entornos altamente dinámicos y distribuidos. Además, el auge de la analítica predictiva y el big data ha permitido a las herramientas como Elasticsearch y Splunk analizar grandes volúmenes de datos generados por sistemas para identificar patrones y anomalías antes de que afecten a los usuarios finales.

(camillepack, 2024)

**Monitoreo de Contenedores:** Con la proliferación de tecnologías de contenedorización como Docker y Kubernetes, el monitoreo de contenedores se ha convertido en una necesidad para asegurar el rendimiento y la disponibilidad de las aplicaciones modernas. Herramientas especializadas como Prometheus ganado popularidad debido a su capacidad para escalar y adaptarse a entornos dinámicos de contenedores. Estas plataformas permiten la recolección de métricas detalladas sobre la salud y el estado de los contenedores, ofreciendo paneles visuales y alertas personalizadas que facilitan la identificación y resolución rápida de problemas de rendimiento.

Estas tendencias reflejan un movimiento hacia soluciones más inteligentes, ágiles y automatizadas en el monitoreo de rendimiento, apoyadas por tecnologías avanzadas como la IA, el ML y la optimización para entornos específicos como la nube y los contenedores. La continua innovación en este campo promete mejorar la eficiencia operativa y la experiencia del usuario al anticipar y mitigar proactivamente los problemas de rendimiento antes de que afecten negativamente a las operaciones empresariales.

## 1.2. Monitoreo de Software

El monitoreo de software se refiere al proceso de supervisar continuamente las aplicaciones y servicios para asegurar que funcionen correctamente y con el rendimiento esperado. Incluye la detección de errores, análisis de rendimiento y la optimización del uso de recursos. Herramientas como Microsoft 365 implementan sistemas de monitoreo avanzado que permiten identificar y solucionar problemas antes de que afecten a los usuarios.

## 1.3. Rendimiento de Software

El rendimiento de software es una medida de la eficiencia con la que un programa o sistema opera. Esto incluye el tiempo de respuesta, la capacidad de procesamiento y la utilización de recursos. En el contexto de Microsoft 365, la implementación de herramientas de IA, como Copilot, ha demostrado mejorar el rendimiento al automatizar tareas repetitivas y proporcionar asistencia en tiempo real.



#### 1.4. Integración de IA en Microsoft 365

La inteligencia artificial ha revolucionado la manera en que las aplicaciones de productividad funcionan. Microsoft Copilot es un ejemplo destacado de esta integración, utilizando IA generativa para asistir a los usuarios en diversas tareas. Estudios han mostrado que la integración de IA en herramientas de productividad puede aumentar la eficiencia y precisión del trabajo, ofreciendo soluciones y recomendaciones basadas en datos en tiempo real.

#### 1.5. Impacto de Copilot en el Monitoreo y Rendimiento

Copilot de Microsoft 365 ha tenido un impacto significativo en la forma en que las organizaciones monitorean y gestionan el rendimiento de sus aplicaciones. La IA permite una supervisión más detallada y precisa, identificación de problemas antes de que se conviertan en críticos y optimización continua de recursos. Esto se traduce en un mejor rendimiento y una mayor satisfacción del usuario.

#### 1.6. Mantenimiento de Software

El mantenimiento de software se refiere a las actividades necesarias para corregir, actualizar y mejorar el software después de su implementación inicial. El objetivo del mantenimiento es asegurar que el software siga siendo útil, eficiente y libre de errores a lo largo del tiempo. Las actividades de mantenimiento se pueden categorizar en cuatro tipos principales:

1. **Correctivo:** Consiste en identificar y corregir errores o defectos en el software que se descubren después de su lanzamiento. Esto incluye la resolución de fallos que afectan el rendimiento y la funcionalidad.
2. **Adaptativo:** Implica modificar el software para que pueda funcionar en un entorno informático nuevo o cambiante, como la actualización de sistemas operativos, hardware o software de soporte.
3. **Perfectivo:** Se refiere a la mejora de las funcionalidades del software, añadiendo nuevas capacidades o mejorando las existentes para satisfacer mejor las necesidades de los usuarios.
4. **Preventivo:** Involucra cambios realizados para detectar y corregir problemas potenciales antes de que se conviertan en errores serios. Esto incluye refactorización de código y optimización para prevenir futuros problemas.

## Rendimiento de Software

El rendimiento de software se refiere a la eficiencia con la que un sistema o aplicación informática ejecuta sus funciones y responde a las solicitudes de los usuarios. El rendimiento puede medirse a través de diversos parámetros, incluyendo:

**Tiempo de Respuesta:** Es el tiempo que tarda el software en responder a una solicitud del usuario o de otro sistema. Un menor tiempo de respuesta indica un mejor rendimiento.

**Capacidad de Procesamiento:** La cantidad de datos o número de transacciones que el software puede manejar en un periodo de tiempo determinado. Un software con alta capacidad de procesamiento puede gestionar grandes volúmenes de datos de manera eficiente.

**Utilización de Recursos:** Hace referencia al uso eficiente de los recursos del sistema, como CPU, memoria y almacenamiento. Un buen rendimiento implica que el software utiliza estos recursos de manera óptima, sin sobrecargar el sistema.

**Escalabilidad:** La capacidad del software para mantener o mejorar su rendimiento a medida que se incrementa la carga de trabajo. Un software escalable puede manejar un mayor número de usuarios o un volumen creciente de datos sin degradar su rendimiento.

**Estabilidad:** Se refiere a la capacidad del software para funcionar sin fallos durante un periodo prolongado. Un software estable tiene menos errores y caídas, lo que contribuye a un mejor rendimiento general.

Microsoft Copilot para Microsoft 365 es una herramienta de productividad con tecnología de inteligencia artificial que coordina modelos de lenguaje grande (LLM), contenido en Microsoft Graph y las aplicaciones de Microsoft 365 que se usan todos los días, como Word, Excel, PowerPoint, Outlook, Teams y otros. Esta integración proporciona asistencia inteligente en tiempo real, lo que permite a los usuarios mejorar su creatividad, productividad y capacidades.

Copilot para Microsoft 365 usa una combinación de LLM, un tipo de algoritmo de inteligencia artificial (IA) que usa técnicas de aprendizaje profundo y grandes conjuntos de datos para comprender, resumir, predecir y generar contenido.

Estos LLM incluyen modelos previamente entrenados, como transformadores pre entrenados (GPT) generativos, como GPT-4, diseñados para destacar en estas tareas.

### Integración de Copilot con Graph y Aplicaciones Microsoft 365

Microsoft Copilot para Microsoft 365 es un sofisticado motor de procesamiento y orquestación que proporciona funcionalidades de productividad con tecnología de IA mediante la coordinación de los siguientes componentes:

#### Modelos de lenguaje grandes (LLM)

Contenido de Microsoft Graph como correos electrónicos, chats y documentos a los que tiene permiso para acceder.

Las aplicaciones de productividad de Microsoft 365 que se usan todos los días, como Word y PowerPoint.

Las aplicaciones de Microsoft 365 (como Word, Excel, PowerPoint, Outlook, Teams y Loop) funcionan con Copilot para dar soporte a los usuarios en el contexto de su trabajo.

#### ¿Cómo funciona Microsoft Copilot para Microsoft 365?

Las funcionalidades de Microsoft Copilot para Microsoft 365 que los usuarios ven en Aplicaciones de Microsoft 365 y otras superficies aparecen como características inteligentes, funcionalidad y capacidad de consulta. Nuestros LLM básicos y las tecnologías de Microsoft patentadas funcionan en conjunto en un sistema subyacente que ayuda a acceder, usar y gestionar de forma segura los datos de la organización.

- Las Aplicaciones de Microsoft 365 (como Word, Excel, PowerPoint, Outlook, Teams y Loop) funcionan junto con Copilot para Microsoft 365 para ayudar a los usuarios con su trabajo. Por ejemplo, Copilot en Word se ha diseñado para ayudar a los usuarios durante el proceso de creación, comprensión y edición de documentos en concreto. De forma similar, Copilot en las otras aplicaciones ayuda a los usuarios en el contexto de su trabajo dentro de esas aplicaciones.
- Microsoft Copilot con chat basado en Graph le permite llevar el contenido y el contexto del trabajo a las funcionalidades de chat de Microsoft Copilot. Con el chat

- basado en Graph, puede redactar contenido, ponerse al día de lo que ha perdido y obtener respuestas a las preguntas a través de mensajes abiertos, todo ello de forma segura en los datos de su trabajo. Use Copilot basado en Graph en muchas superficies, incluido Microsoft Teams, en Microsoft365.com y en copilot.microsoft.com.
- Microsoft Graph ha sido fundamental durante mucho tiempo para Microsoft 365. Incluye información sobre las relaciones entre los usuarios, las actividades y los datos de la organización. La API de Microsoft Graph aporta contexto adicional de las señales de los clientes para la consulta, como
- información de correos electrónicos, chats, documentos y reuniones. Para obtener más información, consulte
- Información general sobre Microsoft Graph y Principales servicios y características de Microsoft Graph.
- Semantic Index para Copilot utiliza múltiples LLM que se asientan sobre Microsoft Graph para interpretar las consultas de los usuarios y producir respuestas sofisticadas, significativas y multilingües que le ayuden a ser más productivo. Le permite buscar rápidamente a través de miles de millones de vectores (representaciones matemáticas de características o atributos) para poder conectar con la información más relevante y procesable de la organización.

En el diagrama siguiente se proporciona una representación visual de cómo funciona Microsoft Copilot para Microsoft 365.

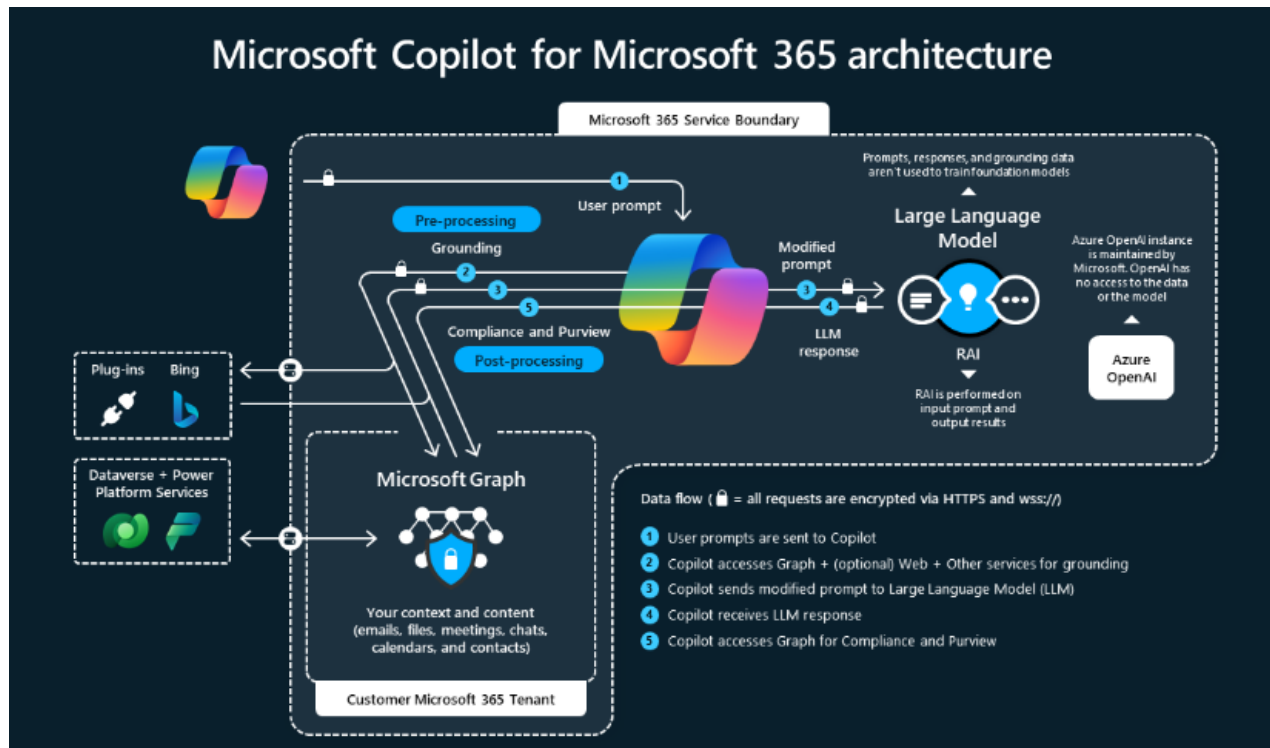


Ilustración 1 copilot

<https://learn.microsoft.com/es-es/copilot/microsoft-365/microsoft-365-copilot-overview>

A continuación, se explica cómo funciona Microsoft Copilot para Microsoft 365:

Copilot recibe una consulta de entrada de un usuario en una aplicación como Word o PowerPoint.

A continuación, Copilot procesa previamente la consulta a través de un enfoque denominado grounding (nociones), que mejora la especificidad de la consulta, lo que garantiza que se obtienen respuestas relevantes y procesables para la tarea específica. La consulta puede incluir texto de archivos de entrada u otro contenido detectado por Copilot, y Copilot envía la consulta al LLM para su procesamiento. Copilot solo accede a los datos a los que un usuario ya tiene acceso, en función, por ejemplo, de los controles de acceso basados en roles de Microsoft 365 existentes.

Copilot toma la respuesta del LLM y posteriormente la procesa. Este procesamiento posterior incluye otras llamadas de grounding a Microsoft Graph, comprobaciones de inteligencia artificial responsable, revisiones de seguridad, cumplimiento y privacidad, y generación de comandos. (camillepack, 2024)

Copilot devuelve la respuesta a la aplicación, donde el usuario puede revisar y evaluar la respuesta.

Nos referimos a la solicitud del usuario y a la respuesta de Copilot a esa solicitud como el "contenido de las interacciones" y el registro de esas interacciones es el historial de interacciones de Copilot del usuario.

Microsoft Copilot para Microsoft 365 procesa y organiza de forma iterativa estos sofisticados servicios para poder generar resultados relevantes para la organización, ya que se basan contextualmente en los datos organizativos.

### Confiabilidad y seguridad

Conforme los sistemas empresariales se insertan profundamente en las vidas empresariales y personales, se incrementan los problemas que derivan las fallas del sistema y del software.

Una falla del software del servidor en una empresa de comercio electrónico podría conducir a dicha compañía hacia una gran pérdida de ingresos y, posiblemente también de clientes. un error de software en un sistema de software en un sistema de control embebido en un automóvil provocaría costosas devoluciones de ese modelo de reparación y, en los peores casos, sería un factor que contribuya a los accidentes.

La infección de la compañía del pc con malware requiere costosas operaciones de limpieza para solventar el problema y quizá de como resultado la pérdida o el daño de información sensible.

Puesto que los sistemas intensivos de software son tan importantes para los gobiernos, la compañía y los individuos, es esencial que este tipo de software sea digno de confianza. El software debe estar disponible cuando se requiera y ejecutarse correctamente y sin efectos colaterales indeseables, como la circulación de información no autorizada.

La confiabilidad de los sistemas es ahora más importante que su funcionalidad detallada por las sigs. Funcionalidades.

1. las fallas del sistema afectan a un gran número de individuos muchos sistemas incluyen funcionalidad que se usa rara a la vez. Si esta funcionalidad se retirara del sistema, tan solo un número menor de usuarios resultarían afectados. Las fallas del sistema, que afectan la disponibilidad de un sistema, gravitarían potencialmente a todos los usuarios del sistema. La falla significaría que es imposible continuar con la normalidad del negocio.

2. los usuarios rechazan a menudo los sistemas que son poco fiables, carecen de protecciones o son inseguros si los usuarios descubren que un sistema es poco fiables o inseguro, lo rechazarán, más aún, ellos también pueden negarse adquirir o usar otros productos de la compañía que desarrolló el sistema que no es fiable, porque consideran que dichos productos tienen la misma probabilidad de no ser fiables o seguros.

3. los costos por las fallas del sistema suelen ser enormes para ciertas aplicaciones, como un sistema de control de reactor o un sistema de navegación de aeronaves, el costo por la falla del sistema es una orden de magnitud mayor que el costo del sistema de control.

4. los sistemas no confiables pueden causar pérdida de información los datos son muy costosos de recolectar y mantener; por lo general, valen mucho más que el sistema de cómputo donde se procesan. El costo por recuperar datos perdidos o contaminados generalmente es muy alto



# Tecnologías a implementar

## Copilot 365

Copilot 365 es una herramienta de asistencia basada en inteligencia artificial integrada en el entorno de desarrollo de Visual Studio y Visual Studio Code. Proporciona sugerencias de código y configuraciones basadas en el contexto del proyecto.

### Uso en el Proyecto:

**Creación de Scripts de Monitoreo:** Copilot 365 asistirá en la escritura de scripts de monitoreo en Python y Bash, sugiriendo fragmentos de código y funciones completas que recolecten métricas de los sistemas y aplicaciones.

**Configuración de Dashboards y Alertas:** Ayudará a crear y configurar archivos JSON y YAML necesarios para los dashboards de las alertas de Prometheus y Alertmanager.

**Automatización de Tareas:** Facilitará la automatización de tareas rutinarias mediante sugerencias inteligentes y completado automático de comandos y configuraciones.

## Prometheus

**Descripción:** Prometheus es una herramienta de monitoreo de código abierto diseñada para recolectar y almacenar métricas como series temporales. Es ampliamente utilizada por su robustez y facilidad de integración con otros sistemas.

### Uso en el Proyecto:

**Recolección de Métricas:** Prometheus recolecta métricas de diversos servicios y aplicaciones mediante exporters. Los exporters son componentes que se instalan en los sistemas a monitorear y exponen métricas en un formato que Prometheus puede recolectar.

**Almacenamiento de Datos:** Las métricas recolectadas se almacenan en la base de datos de series temporales de Prometheus.

Consulta de Datos: Utiliza el lenguaje de consulta PromQL para analizar y extraer información de las métricas almacenadas.

Alertas: Prometheus puede configurarse para generar alertas basadas en condiciones específicas definidas por el usuario. Estas alertas se gestionan mediante Alertmanager, un componente de Prometheus para el manejo de notificaciones.

## Elastic Stack (ELK)

Elastic Stack, comúnmente conocido como ELK Stack, es un conjunto de herramientas que incluye Elasticsearch, Logstash y Kibana para la búsqueda, análisis y visualización de datos en tiempo real.

Elasticsearch: Un motor de búsqueda y análisis distribuido que almacena y permite buscar a través de grandes volúmenes de datos.

Logstash: Una herramienta de procesamiento de datos que puede ingerir datos de diversas fuentes, transformarlos y enviarlos a un destino como Elasticsearch.

Kibana: Una herramienta de visualización que se integra con Elasticsearch y permite crear dashboards interactivos y realizar análisis de datos.

## Uso en el Proyecto:

Recolección y Almacenamiento de Logs: Logstash procesa y envía logs a Elasticsearch, donde se almacenan y se indexan para una búsqueda rápida.

Análisis y Búsqueda: Elasticsearch almacena los datos recolectados y permite realizar búsquedas y análisis sobre estos datos.

Visualización: Kibana proporciona una interfaz para visualizar los datos almacenados en Elasticsearch, facilitando la creación de dashboards y la realización de análisis detallados.

WMI: Windows Management Instrumentation (WMI) es un marco de herramientas del sistema que se ha integrado en el sistema operativo Windows. Se puede acceder a WMI a través de las redes y permite a los usuarios consultar los sistemas remotos para obtener información sobre ellos. WMI es potente y flexible, y si se usa correctamente puede suministrar la información más importante sobre sus computadoras, servidores y equipos virtuales de Hyper-V.

Windows Management Instrumentation (WMI) es una potente herramienta de gestión y supervisión del sistema operativo Windows. Proporciona una arquitectura flexible y extensible para acceder a datos de gestión y automatizar tareas administrativas. Tanto si gestiona una pequeña red de ordenadores como una gran red empresarial, las herramientas WMI pueden ayudarle a agilizar sus operaciones y mejorar su eficacia.

Con los conocimientos y las herramientas adecuadas, puede aprovechar WMI al máximo. Tanto si utiliza la línea de comandos de WMI, como si escribe consultas de WMI o crea scripts de WMI, existe una gran cantidad de recursos disponibles para ayudarle a sacar el máximo partido de WMI. Y con el desarrollo continuo del sistema operativo Windows y WMI, puede esperar características y capacidades aún más potentes en el futuro.

Arquitectura de WMI: La arquitectura WMI es una arquitectura flexible y extensible diseñada para proporcionar una forma unificada de acceder a la información de gestión en un entorno empresarial. En el corazón de la arquitectura WMI se encuentran los proveedores WMI. Los proveedores WMI actúan como intermediarios entre el servicio WMI y los objetos gestionados en el Modelo de controlador de Windows.

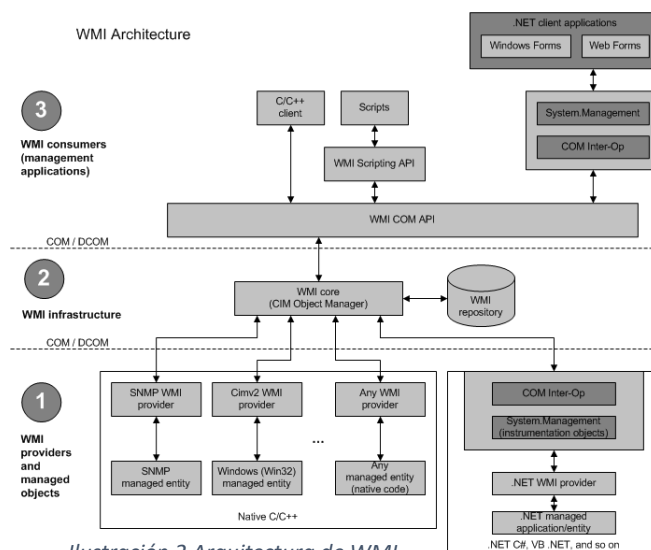


Ilustración 2 Arquitectura de WMI

Trabajar con WMI: Trabajar con WMI implica utilizar la línea de comandos WMI, el Lenguaje de Consulta WMI (WQL) y los scripts WMI. La herramienta de línea de comandos WMI, también conocida como WMIC, es una interfaz de línea de comandos para la API WMI. Permite a los administradores realizar varias tareas de gestión del sistema desde el símbolo del sistema. Los scripts WMI se utilizan para automatizar tareas administrativas y manipular datos de gestión.

Consultar datos WMI: Las consultas de datos son el tipo más común de consulta WMI Query Language (WQL). Se utilizan para recuperar datos del repositorio WMI. Las consultas de datos pueden utilizarse para recuperar instancias de una clase específica o para recuperar instancias que cumplan ciertos criterios.

Por ejemplo, una consulta de datos puede recuperar todas las instancias de la clase Win32\_Process (que representa los procesos que se ejecutan en un sistema Windows), o puede recuperar sólo aquellas instancias en las que la propiedad Name sea "notepad.exe".

Prometheus: es un software especializado como sistema de monitorización y alertas escrito en el lenguaje de programación Go. Todos los datos y métricas se almacenan en la base de datos como series temporales (junto al instante de tiempo en el que el valor se ha registrado). También es posible añadir etiquetas de tipo clave-valor junto a estas métricas.

Las métricas que almacena Prometheus pueden ser de cualquier tipo, y dependerán de la naturaleza de la aplicación o del sistema que se quiera monitorizar. Por ejemplo, puede ser el uso de CPU o de memoria, número de conexiones, número de peticiones o cantidad de sesiones activas. Todas las mediciones y métricas recogidas ayudarán a diagnosticar errores o problemas de servicio en los sistemas y aplicaciones que se monitorizan.

Arquitectura de Prometheus:

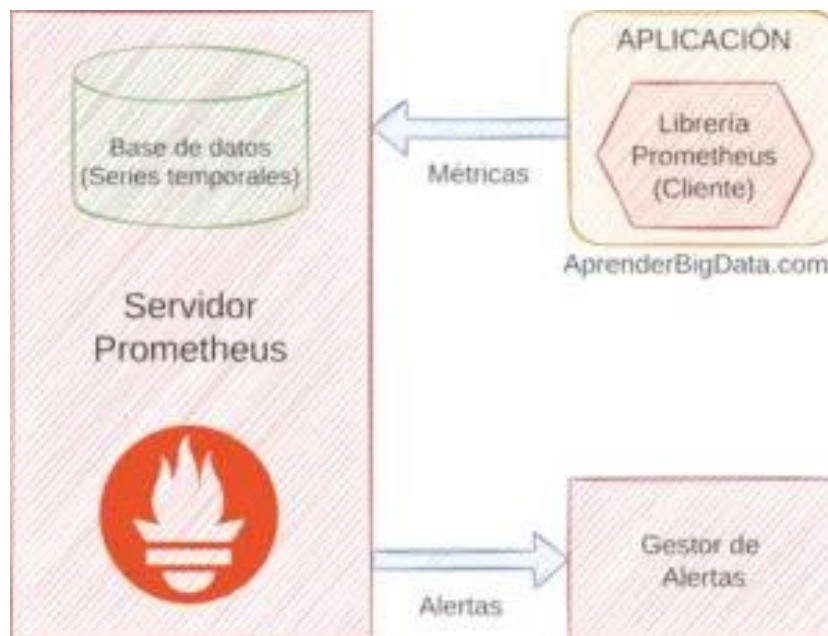
Prometheus tiene 3 componentes principales.

Servidor: Almacena los datos de las métricas.

Librería Cliente: Se usa para calcular y exponer las métricas al cliente.

Gestor de alertas: Genera alertas basadas en reglas.

Para monitorizar una aplicación deberemos configurar con código la librería de cliente de forma que exponga la métrica que queremos registrar. Ahora que la aplicación expone nuestra métrica, ejecutaremos el servidor de Prometheus y lo configuraremos para extraerla de la aplicación con unos intervalos de tiempo establecidos.



*Ilustración 3 Arquitectura de Prometheus*

**Contador:** El tipo de métrica contador es un valor que solo se puede incrementar o bien resetear. Se puede usar para contar el número de peticiones o de errores en una aplicación, que son métricas que nunca se reducen.

**Gauge o medidor:** es un valor numérico que puede incrementarse o decrementar. Un ejemplo puede ser el número de servidores en un sistema distribuido.

**Histograma:** representa los valores en agrupaciones predefinidas y acumuladas en el tiempo. Por ejemplo, se puede usar un histograma para medir los tiempos de respuesta de nuestra aplicación en los intervalos de tiempo establecidos para cada petición de un cliente.

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

## Herramientas de Alertas

Estas herramientas se utilizan para gestionar alertas y notificaciones, asegurando que las anomalías detectadas en el sistema sean comunicadas rápidamente a los responsables.

PagerDuty: Una plataforma de gestión de incidentes que ayuda a gestionar alertas y coordinar la respuesta a incidentes.

Opsgenie: Similar a PagerDuty, se centra en la gestión de alertas y la coordinación de respuestas a incidentes.

Slack/Email: Herramientas de notificación en tiempo real que permiten enviar alertas a través de mensajes directos o correos electrónicos.

Uso en el Proyecto:

Integración con Alertmanager Prometheus se configura para enviar alertas a estas herramientas mediante Alertmanager. Las notificaciones pueden enviarse a PagerDuty, Opsgenie, Slack o vía email para garantizar que los equipos responsables sean informados de inmediato sobre cualquier problema.

### Lenguajes de Programación y Scripting

El desarrollo de scripts y automatizaciones se realiza utilizando lenguajes de programación y scripting adecuados para la recolección de datos, procesamiento y configuración del sistema

Python: Utilizado para desarrollar scripts de monitoreo, análisis de datos y automatización de tareas.

Bash: Utilizado para scripts de administración del sistema y tareas de monitoreo básicas.

JavaScript: es un lenguaje de secuencias de comandos que te permite crear contenido de actualización dinámica, controlar multimedia, animar imágenes y prácticamente todo lo demás. (Está bien, no todo, pero es sorprendente lo que puedes lograr con unas pocas líneas de código JavaScript).

Uso en el Proyecto:

**Automatización y Scripts:** Los scripts desarrollados en Python y Bash se utilizan para recolectar y procesar métricas, interactuar con APIs y automatizar tareas rutinarias.

**Configuración y Personalización:** JavaScript se utiliza para personalizar dashboards y desarrollar extensiones que mejoren la funcionalidad de las herramientas de visualización.

## Implementación del Proyecto

Prometheus se configura para recolectar métricas de los sistemas y aplicaciones a través de diferentes exporters. La configuración incluye definir los jobs y targets que Prometheus monitoreará.

### Configuración de Elastic Stack

Logstash se configura para recolectar logs de las aplicaciones y sistemas, procesarlos y enviarlos a Elasticsearch. Elasticsearch almacena estos logs, permitiendo su análisis y búsqueda rápida. Kibana se utiliza para crear dashboards que visualizan estos logs y facilitan el análisis.

### Configuración de Alertmanager

Alertmanager se configura para gestionar las alertas generadas por Prometheus. Las reglas de alerta se definen para detectar condiciones específicas y enviar notificaciones a herramientas de gestión de alertas y notificaciones como PagerDuty, Opsgenie, Slack o Email.

## Integración y Despliegue

El proyecto se despliega utilizando Docker y Docker Compose para simplificar la gestión y escalabilidad de los diferentes componentes. Cada servicio (Prometheus, Elasticsearch, Logstash) se define en un archivo ``docker-compose.yml``, facilitando su despliegue y administración.

(service, 2023) <https://www.soluciones-im.com/es/la-monitorizacion-de-sistemas-informaticos-en-8-pasos>



## Métricas de para medir el rendimiento de nuestras aplicaciones

### CPU

La utilización de la Unidad Central de Procesamiento (CPU) incide directamente en la capacidad de respuesta de una aplicación. Los momentos de intensa actividad en el uso de la CPU pueden apuntar hacia diversas problemáticas. Más específicamente, esto sugiere que la aplicación está ocupada dedicando tiempo a operaciones computacionales, lo cual deriva en una merma en la capacidad de respuesta. Los episodios de alta utilización deben ser interpretados como un inconveniente de rendimiento, dado que indican que la CPU ha llegado a su límite de utilización

### Memoria

La gestión de la memoria emerge como un indicador esencial del rendimiento de la aplicación. La utilización elevada de la memoria denota una demanda sustancial de recursos en el servidor. Al realizar un seguimiento del uso de la memoria de una aplicación, preste atención a la frecuencia de fallos de página y los tiempos de acceso al disco. Si ha asignado una cantidad inapropiada de memoria virtual, su aplicación dedica más tiempo a la híper-paginación que a cualquier otra operación.

### Solicitudes por minuto y bytes por solicitud

La supervisión de la frecuencia de solicitudes recibidas por minuto en la API de su aplicación se revela crucial para evaluar el rendimiento del servidor ante diversas cargas. Asimismo, es imperativo llevar un detallado registro de la cantidad de datos manejada por la aplicación en cada solicitud. Podría constatar que la aplicación recibe más peticiones de las que puede gestionar, o que la magnitud de los datos a procesar afecta negativamente su rendimiento.

### Latencia y tiempo de actividad

La latencia —comúnmente medida en milisegundos— hace referencia al intervalo entre la acción de un usuario en una aplicación y la subsiguiente respuesta de la aplicación. Un incremento en la latencia incide de manera directa en el tiempo de carga de la aplicación. Se aconseja hacer uso de un servicio de ping para evaluar

la continuidad temporal. Estos servicios pueden ser programados para ejecutarse en intervalos específicos, permitiendo así verificar el estado operativo de una aplicación.

## Seguridad

Resulta esencial garantizar la seguridad tanto de su aplicación como de sus datos. Evalúe la amplitud de la aplicación protegida por técnicas de seguridad y cuánta queda expuesta y desprotegida. Asimismo, es fundamental contar con un plan para calcular el tiempo que demanda —o podría demandar— abordar vulnerabilidades específicas de seguridad.

Satisfacción del usuario/puntuaciones de Apdex.

El Índice de Rendimiento de Aplicaciones (Apdex) se presenta como una normativa abierta destinada a evaluar los tiempos de respuesta en aplicaciones web, comparándolos con un umbral predefinido. Su cálculo se basa en la proporción entre tiempos de respuesta satisfactorios e insatisfactorios. El tiempo de respuesta se refiere al intervalo que demanda un recurso para ser entregado al solicitante tras su requerimiento.

Ilustrativamente, imagine que ha establecido un umbral temporal designado como T. En consecuencia, todas las respuestas completadas en T o menos se consideran satisfactorias para el usuario, mientras que aquellas que superan los T segundos se categorizan como insatisfactorias.

## Tiempo medio de respuesta

El cálculo del tiempo de respuesta promedio implica la obtención del promedio de los tiempos de respuesta de todas las solicitudes durante un intervalo de tiempo definido. Un tiempo de respuesta promedio reducido sugiere un rendimiento superior, indicando que la aplicación o el servidor han gestionado las solicitudes o entradas de manera más eficiente.

La determinación del tiempo medio de respuesta se alcanza al dividir el tiempo necesario para responder a las solicitudes en un periodo específico por el total de respuestas durante ese mismo periodo.

## Errores

Este indicador de rendimiento cuantifica el porcentaje de solicitudes que experimentan errores en relación con el total de solicitudes durante un intervalo de tiempo definido. Un incremento en esta proporción sugiere la probabilidad inminente de detectar una falla significativa.

Para realizar un seguimiento de las fallas de la aplicación, puede utilizar los siguientes indicadores:

Excepciones registradas

Excepciones lanzadas

Porcentaje de error HTTP

Básicamente, puede emplear las tasas de error para monitorear cuán frecuentemente su aplicación enfrenta fallas en tiempo real. Además, puede estar alerta a esta métrica de rendimiento para identificar y corregir errores de manera ágil, evitando así inconvenientes que puedan resultar en la caída completa de su sitio.

Recolección de basura

La recolección de elementos no utilizados puede dar lugar a breves interrupciones en su aplicación durante el ciclo de GC. Además, también puede consumir numerosos ciclos de CPU, por lo que es crucial evaluar minuciosamente el rendimiento de la recolección de elementos no utilizados en su aplicación.

Mangos de GC. Esta métrica cuantifica el total de referencias a objetos generadas en una aplicación.

Porcentaje de tiempo dedicado a GC. Este porcentaje refleja el tiempo invertido en la recolección de elementos no utilizados desde el último ciclo de GC.

Tiempo de pausa durante la recolección de basura. Mide la duración en la cual la aplicación se detiene por completo durante un ciclo de GC. Puede minimizarse limitando la cantidad de objetos que deben ser marcados, es decir, objetos candidatos para la recolección de elementos no utilizados.

Eficiencia en la recolección de basura. Indica el porcentaje de tiempo total que la aplicación no ha dedicado a la recolección de elementos no utilizados.

Tasa de creación/recuperación de objetos. Es una medida de la velocidad a la que se generan o recuperan instancias en una aplicación. Cuanto mayor sea la tasa de creación de objetos, más frecuentes serán los ciclos de GC y, por ende, aumentará la utilización de la CPU.

#### Requisitos de velocidad y rendimiento

La velocidad de respuesta a solicitudes es un indicador crucial que proporciona datos sobre las fluctuaciones en el tráfico experimentado por su aplicación. En términos simples, suministra información sobre los periodos de calma y los momentos de mayor actividad cuando su aplicación enfrenta picos de tráfico. Puede establecer conexiones entre las velocidades de respuesta a solicitudes y otras métricas de rendimiento de la aplicación para comprender cómo ajustar la escala de su aplicación. Además, es fundamental supervisar la cantidad de usuarios concurrentes en su aplicación.

Una de las piezas fundamentales para el buen crecimiento de una empresa es optimizar de mejor manera los recursos, ya que de esta forma el ahorro obtenido se traduce en finanzas saludables y un mayor margen de inversión, lo que da como resultado el éxito empresarial.

Si quieres conocer más sobre cómo optimizar los recursos, los beneficios de implementar estrategias para eficientar los procesos, te compartimos 10 consejos para mejorar en este rubro y que, sin importar el tipo de negocio, comience a crecer de forma natural.

Una de las piezas fundamentales para el buen crecimiento de una empresa es optimizar de mejor manera los recursos, ya que de esta forma el ahorro obtenido se traduce en finanzas saludables y un mayor margen de inversión, lo que da como resultado el éxito empresarial.

Si quieres conocer más sobre cómo optimizar los recursos, los beneficios de implementar estrategias para eficientar los procesos, te compartimos 10 consejos para mejorar en este rubro y que, sin importar el tipo de negocio, comience a crecer de forma natural.

Gracias a la digitalización, las demandas en constante evolución y la volatilidad del mercado, entre otros factores, el sector empresarial está experimentando un importante cambio de paradigma, por lo que las organizaciones deben mantenerse al día con las tendencias del mercado a fin de que logren construir una fuerza laboral lista para el futuro.

Una de estas tendencias es la optimización de recursos, sobre lo cual le explicaremos qué es, qué es posible optimizar dentro de un negocio, sus beneficios y 8 formas de optimizar los recursos de su empresa.

La optimización de software es una parte fundamental dentro de cualquier desarrollo. Todo, desde la aplicación más simple hasta el más sofisticado sistema de análisis de big data, se ejecuta sobre recursos de hardware limitados. Por eso, una correcta optimización permitirá que esos recursos se utilicen de forma más eficiente. ¿Quieres saber cómo puedes conseguirla?

Lo primero que tienes que saber es que, al contrario de lo que a veces puede parecer, el proceso de optimización de código no es ajeno ni complementario al desarrollo. Se trata de una parte integral de este. Podemos dividir el proceso de optimización en varios niveles, que a continuación te detallamos.

¿Qué es la optimización de recursos?

La optimización de recursos es una práctica de gestión empresarial diseñada para impulsar la eficiencia, productividad y rendimiento de una organización.

Para alcanzar estos objetivos, abarca procesos como la planificación, pronóstico, asignación y programación de recursos disponibles, utilizándolos inteligentemente para cumplir con los objetivos de la organización.

¿Qué se puede optimizar dentro de una empresa?

Dentro de una empresa se pueden optimizar distintos tipos de recursos:

Recursos humanos: este tipo engloba a todos los colaboradores que brindan sus servicios profesionales en las distintas áreas de una empresa y los que están por ingresar por medio del proceso de reclutamiento.

Recursos financieros: se puede optimizar la capacidad de endeudamiento del negocio, la cantidad de efectivo disponible, las relaciones con inversionistas que financian las actividades y operaciones, entre otros.

Recursos tecnológicos: es posible optimizar recursos tecnológicos como los sistemas informáticos, herramientas, servidores, maquinaria de producción, sistemas de seguridad, entre otros.

Recursos materiales: en esta categoría se incluyen todos los bienes tangibles de una empresa, como propiedades, maquinaria, flota vehicular, herramientas de trabajo y materias primas.

### Uso del Usuario

Los usuarios de herramientas de monitoreo y rendimiento de software pueden incluir:

Desarrolladores de software: Para identificar y corregir problemas de rendimiento durante el desarrollo.

Equipos de operaciones de TI: Para mantener el software funcionando sin interrupciones y resolver rápidamente cualquier problema que surja.

Gestores de proyectos: Para asegurar que los proyectos de software cumplen con los requisitos de rendimiento y calidad.

Empresas: Para optimizar la eficiencia operativa y asegurar una experiencia de usuario positiva.

### 3 estrategias para implementar procesos de mejora en tu empresa

#### 1. Encontrar los indicadores clave de cada proceso de la empresa

Para iniciar la mejora de procesos es necesario identificar todos los procesos que conforman la organización y conocer los indicadores de resultados de cada uno.

El objetivo es encontrar la información histórica de otras acciones que fueron realizadas anteriormente y que generaron buenos resultados para aplicarlas en el

presente. Sin embargo, deberás adaptarlas al contexto actual y a la metodología de gestión que estés llevando a cabo.

## 2. Diseñar la estrategia, escoger los objetivos e identificar qué herramientas digitales serán útiles

Además de identificar cuáles son las áreas y los procesos que necesitan ser optimizados, también es importante que la empresa defina qué objetivos son los que busca alcanzar. Ya sea mejorar sus ventas, el servicio al cliente, aumentar la productividad de sus empleados para que realicen su trabajo en menos tiempo, incrementar su producción, etc.

Este es un paso importante con el que también se escogerán los recursos de software que ayudarán a tu empresa con el proceso de la medición de resultados, para digitalizar procesos administrativos, reducir recursos y tiempos, eliminar procesos manuales, mejorar los servicios de ventas, y cualquier otra necesidad de la empresa.

## 3. Analizar las métricas y los resultados obtenidos continuamente

Finalmente, es necesario analizar las métricas y los resultados obtenidos a partir de la implementación de los procesos de mejora continua. De modo que se pueda conocer si se han alcanzado los objetivos o si es necesario modificar la estrategia.

Asimismo, es importante controlar y darle un seguimiento constante a todos los procesos para identificar cuando se presente un cambio y establecer nuevamente una estrategia que ayude a cumplir los objetivos de la empresa.

¿En qué consiste el despliegue de software?

El despliegue de software es el proceso de entregar aplicaciones de software desde entornos de desarrollo a producción, poniéndolas a disposición de los usuarios finales. Debe garantizar que el software se configure, pruebe, lance e instale de forma coherente y controlada.

¿Por qué el despliegue de software es importante?

El despliegue de software resulta esencial para garantizar que las aplicaciones se entregan e instalan de forma correcta y eficaz. Algunas ventajas que te aporta este proceso son:



Agiliza los tiempos de comercialización: un despliegue veloz ayuda a satisfacer las demandas de los clientes y a adelantarse a la competencia.

Mejora la calidad: también garantiza que la aplicación se entrega en la configuración deseada, con todas las dependencias y ajustes necesarios del sistema. Esto contribuye a minimizar los errores, optimizar el rendimiento y la experiencia del usuario.

Incrementa la seguridad: el proceso garantiza que la aplicación de software está protegida frente a vulnerabilidades que podrían ser aprovechadas por atacantes.

Optimiza el control de los costos: al automatizar el proceso de despliegue, las organizaciones reducen los errores manuales, ahorran tiempo y bajan los gastos.

### Métodos de despliegue de software

Despliegue manual: normalmente lo hace un equipo de profesionales de IT. Se trata de una metodología que puede llevar mucho tiempo y ser propensa a equivocaciones humanas.

1. Despliegue mediante scripts: estos últimos se utilizan para automatizar el proceso, lo cual ahorra tiempo y reduce errores. Pero necesitarás un agente con conocimientos técnicos para configurarlo y mantenerlo. Esta es la vía elegida para hacer el despliegue de software en InvGate Insight.
2. Despliegue continuo: se trata de un método de despliegue automatizado donde el software se entrega continuamente a producción, a menudo con la ayuda de herramientas DevOps. Si bien los tiempos se agilizan y se logra una mayor calidad del software, requiere una inversión significativa en automatización e infraestructura.
3. Despliegue basado en contenedores: es una metodología popular que consiste en empaquetar las aplicaciones en contenedores que permiten desplegarse fácilmente en distintos entornos. Esta vía ayuda a optimizar la portabilidad y la escalabilidad, pero exige cierto nivel de experiencia en la orquestación de los contenedores.
4. Despliegue Azul-Verde: consiste en desplegar dos entornos idénticos, uno “azul” y otro “verde”. Uno se destina a la producción y el otro a probar nuevos cambios. Una vez finalizadas las experimentaciones, las modificaciones se trasladan rápidamente al entorno de producción. Esta vía contribuye a la reducción del tiempo

de inactividad y el riesgo, aunque requiere cierto nivel de automatización y experiencia en orquestación.

5. **Despliegue Rollback:** implica volver a una versión anterior del software en caso de problemas o errores, lo cual ayuda a reducir el período de inactividad y minimizar el impacto de los errores. Para aplicarlo, asegúrate de contar con un sólido sistema de control de versiones.

El proceso de despliegue de software

El proceso de despliegue de software implica una serie de pasos necesarios para entregar e instalar aplicaciones de forma correcta y eficiente. Son los siguientes:

### 1. Planificación

Este paso define el alcance del proceso de despliegue, identifica a las partes interesadas y establece un calendario. A su vez, en esta instancia es esencial asegurar la disponibilidad de los recursos requeridos, como hardware y software. Para todo esto, es aconsejable crear un flujo de trabajo en InvGate Service Desk con la finalidad de garantizar que se siga cada una de las instancias. Asimismo, puedes automatizar acciones en InvGate Insight para ahorrar tiempo y estandarizar procesos.

### 2. Empaquetado

La aplicación de software debe empaquetarse en un formato adecuado para su despliegue. Esto puede implicar crear un instalador o empaquetar la aplicación en una imagen de contenedor.

### 3. Pruebas

Antes de desplegar el software, es importante realizar pruebas exhaustivas para garantizar que la aplicación funciona correctamente y cumple los requisitos de los usuarios finales. Evaluaciones unitarias, de integración y de aceptación del cliente, son algunas de las posibilidades.

#### 4. Configuración

El siguiente paso consiste en configurar la aplicación de software de acuerdo con las necesidades específicas de los usuarios finales. Algunas cuestiones inherentes a esta fase son: la creación de la base de datos y la configuración de los roles de usuario, permisos y ajustes del sistema. Mediante InvGate Insight accedes fácilmente a la información de los activos y asignas la propiedad de la licencia al usuario.

#### 5. Despliegue

En este paso, se produce el despliegue de software en el entorno de producción, lo cual involucraría estas posibilidades: copiar archivos en el servidor, ejecutar scripts o utilizar una herramienta de despliegue. Las integraciones de desktop remoto de InvGate Insight proporcionan un acceso sencillo a los dispositivos donde se va a desplegar.

#### 6. Pruebas posteriores al despliegue

Una vez desplegada la aplicación de software, es importante realizar pruebas posteriores para garantizar que la aplicación funciona correctamente en el entorno de producción.

#### 7. Monitoreo y mantenimiento

También es esencial supervisar el rendimiento de la aplicación de software y realizar tareas de mantenimiento periódicas, como actualizaciones, implementación de parches y copias de seguridad.

#### Definición de Escalabilidad

Cuando hablamos de la escalabilidad de un proyecto o negocio, nos referimos a su capacidad inherente para expandirse o contraerse de manera efectiva para satisfacer las demandas y requerimientos cambiantes que puedan surgir.

En esencia, encarna el potencial de un proyecto o negocio para crecer y adaptarse a una mayor carga de trabajo, nuevas funcionalidades y los desafíos presentados por un entorno dinámico y en constante evolución.

## La importancia de la escalabilidad

La escalabilidad es un factor importante cuando se trata de diseñar e implementar un sistema de software exitoso. Garantizan que el sistema pueda manejar las crecientes demandas de los usuarios, proporcionar una experiencia de usuario fluida y receptiva y adaptarse a los requisitos cambiantes a medida que crece la aplicación.

Manejo del aumento de la demanda: a medida que su aplicación crezca en popularidad y atraiga a más usuarios, sus recursos se verán cada vez más gravados. Para garantizar un funcionamiento eficiente y evitar interrupciones en el servicio, es esencial diseñar software fácilmente escalable que ofrezca un rendimiento confiable bajo cargas elevadas.

Respaldar el crecimiento empresarial: el software escalable y de alto rendimiento puede ayudar a una empresa a expandirse al proporcionar una aplicación que puede adaptarse a volúmenes crecientes de usuarios y a sus diversas necesidades. Esta adaptabilidad permite a una empresa ofrecer más servicios, ingresar a nuevos mercados y ejecutar sus estrategias a largo plazo.

Mejorar la experiencia del usuario: una aplicación con buen rendimiento permitirá a los usuarios realizar sus tareas de manera eficiente y sin demoras frustrantes. Las experiencias positivas de los usuarios pueden aumentar la satisfacción del usuario, lo que genera una mayor participación, una mejor retención de los usuarios y una mejor reputación de la marca.

Mitigar el riesgo y reducir la complejidad: una arquitectura de software escalable y de alto rendimiento puede ayudar a desacoplar componentes, reducir la complejidad del sistema y gestionar el riesgo. Esto puede conducir a un sistema más estable y fácil de mantener, lo que a su vez ayuda a evitar costosas interrupciones o fallas del sistema.

Tipos de escalabilidad: vertical y horizontal

Escalabilidad vertical: La escalabilidad vertical, o "ampliación", implica agregar más recursos a su hardware existente. Esto puede incluir aumentar la potencia de la CPU, la memoria o la capacidad de almacenamiento para adaptarse a las mayores demandas de su aplicación. La escalabilidad vertical se puede lograr mediante:

Actualizar el hardware del servidor, como agregar más RAM, procesadores más rápidos o unidades de estado sólido.

Optimizar su software para utilizar los recursos del sistema de manera más eficiente, como mejorar el rendimiento de las consultas o implementar estrategias de almacenamiento en caché.

### Escalabilidad horizontal

La escalabilidad horizontal, o "ampliación horizontal", se refiere a la expansión de una aplicación agregando más hardware o nodos para distribuir la carga de trabajo. Esto se logra agregando máquinas físicas o virtuales, que trabajan en paralelo para aumentar el poder y la capacidad de procesamiento. La escalabilidad horizontal puede ofrecer varios beneficios:

Permite un crecimiento casi infinito, ya que prácticamente no hay límite para la cantidad de máquinas que se pueden agregar.

Puede mejorar la tolerancia a fallas y la resiliencia de su sistema al distribuir la carga de trabajo entre múltiples nodos, lo que reduce el impacto de una sola falla.

Permite el uso eficiente de los recursos de computación en la nube, lo que permite un escalamiento rentable y bajo demanda.

Factores clave que afectan el rendimiento y la escalabilidad:

Varios factores entran en juego al evaluar el rendimiento y la escalabilidad de una aplicación de software. Familiarizarse con estos factores puede ayudar a los desarrolladores a identificar cuellos de botella, eliminar ineficiencias y optimizar sus sistemas para cumplir con los requisitos en expansión:

Diseño de software: el diseño de software adecuado es fundamental para lograr escalabilidad y rendimiento. Técnicas como la modularización, el desacoplamiento y la separación de preocupaciones pueden ayudar a crear aplicaciones más escalables y mantenibles.

Almacenamiento y recuperación de datos: el manejo eficiente de los datos es esencial para el software escalable. Se pueden emplear varios sistemas de almacenamiento, como bases de datos relacionales, bases de datos NoSQL y

mecanismos de almacenamiento en caché, para optimizar las operaciones de almacenamiento y recuperación de datos.

Redes: la latencia, el ancho de banda y la confiabilidad de la red impactan significativamente el rendimiento del software. El uso de protocolos de red, algoritmos de compresión y redes de entrega de contenido (CDN) adecuados puede ayudar a mitigar los cuellos de botella de la red y mejorar el rendimiento.

Hardware: el hardware subyacente en el que se ejecuta el software, incluidos servidores, almacenamiento y dispositivos de red, puede limitar el rendimiento y la escalabilidad. Las actualizaciones periódicas de hardware y las estrategias de utilización eficiente de los recursos pueden ayudar a abordar estos desafíos.

Patrones de usuario: anticipar el comportamiento del usuario y diseñar el software en consecuencia puede mejorar significativamente el rendimiento. El análisis de los patrones de usuario para identificar funciones utilizadas con frecuencia, tiempos de uso máximo y cuellos de botella comunes puede contribuir a los esfuerzos de optimización del software.

## Importancia del Monitoreo y Rendimiento de Software

Detección temprana de problemas: Ayuda a identificar y solucionar problemas antes de que afecten a los usuarios finales.

Optimización de recursos: Permite utilizar eficientemente los recursos del sistema, reduciendo costos y mejorando el rendimiento. Mejora continua: Proporciona datos que ayudan a mejorar continuamente el software.

Satisfacción del cliente: Un software rápido y confiable mejora la experiencia del usuario, lo que puede aumentar la satisfacción y la retención de clientes.

En los últimos años, se prevé que el monitoreo y rendimiento de software continuará evolucionando gracias a avances como:

Inteligencia Artificial y Machine Learning: Herramientas más avanzadas que pueden predecir problemas antes de que ocurran y optimizar el rendimiento automáticamente.

Monitoreo en tiempo real: Mejoras en la capacidad de monitorear sistemas en tiempo real para una respuesta más rápida a los problemas.

Integración con DevOps: Mayor integración con prácticas de DevOps para un ciclo de desarrollo más ágil y eficiente.

Automatización avanzada: Incremento en la automatización de tareas de monitoreo y mantenimiento, reduciendo la intervención manual.

#### Beneficios Actuales del Monitoreo y Rendimiento de Software

Reducción de tiempos de inactividad: Minimiza el tiempo en que el software no está operativo, mejorando la continuidad del negocio.

Mejora de la eficiencia operativa: Asegura que los recursos del sistema se usen de manera óptima.

Aumento de la productividad: Los equipos pueden identificar y resolver problemas rápidamente, dedicando más tiempo a innovar.

Seguridad mejorada: Monitoreo constante que puede detectar comportamientos anómalos y posibles brechas de seguridad.

Entre 2025 y 2030, el monitoreo y rendimiento de software se verá significativamente potenciado por avances en inteligencia artificial, machine learning, y tecnologías de computación en la nube y en el borde. Estos desarrollos permitirán una optimización continua, una respuesta rápida a problemas y una mejora sustancial en la experiencia del usuario. Las empresas que adopten estas tecnologías podrán ofrecer servicios más eficientes, seguros y adaptables, manteniéndose competitivas en un mercado en constante evolución.

El monitoreo y rendimiento de software es una inversión estratégica para cualquier empresa que desee ofrecer productos y servicios de alta calidad. Los beneficios se traducen en una mejor experiencia para los usuarios y una mayor eficiencia operativa y competitividad para las empresas. Estas herramientas no solo ayudan a mantener el software funcionando sin problemas, sino que también permiten a las empresas adaptarse rápidamente a las necesidades del mercado y mejorar continuamente sus ofertas.

# Metodología

¿Qué es la metodología Prototipó?

La metodología de prototipo es un enfoque integral en el desarrollo de software, caracterizado por la creación iterativa de prototipos funcionales y rápidos que simulan el producto final antes de su implementación completa. Estos prototipos actúan como versiones preliminares de la aplicación, permitiendo a los desarrolladores y usuarios finales evaluar, validar y refinar tanto la funcionalidad como el diseño del software.

La esencia de esta metodología radica en su capacidad para facilitar una retroalimentación continua y directa de los usuarios, lo que posibilita ajustes tempranos y precisos en el desarrollo del producto. A través de sucesivas iteraciones, se pueden identificar y corregir posibles fallos y mejorar las características de usabilidad, garantizando que el producto final cumpla con las expectativas y requisitos del cliente.

Asimismo, la metodología de prototipo favorece una comunicación más efectiva entre el equipo de desarrollo, promoviendo una comprensión mutua de los objetivos y funcionalidades deseadas. Esto no solo optimiza el proceso de desarrollo, sino que también contribuye a la reducción de riesgos y costos asociados a la corrección de errores en etapas avanzadas del ciclo de vida del software.

## Etapas de la metodología Prototipo

### 1. Requerimientos Iniciales

En esta etapa, se identifican los requisitos iniciales del software y se determina el alcance del proyecto. Los desarrolladores trabajan en estrecha colaboración con los usuarios finales y otras partes interesadas para comprender claramente las expectativas y los objetivos del sistema.

### 2. Diseño del Prototipo

En esta etapa, se crea un diseño preliminar del prototipo de software. Se definen



la arquitectura, la interfaz de usuario y las funcionalidades principales del sistema. El diseño se basa en los requisitos iniciales y se presenta a los usuarios finales para su evaluación y retroalimentación.

### 3. Desarrollo del Prototipo

En esta etapa, se implementa el prototipo funcional del software. Los desarrolladores utilizan tecnologías y herramientas adecuadas para crear una versión preliminar de la aplicación que incluye las funcionalidades principales. Este prototipo es lo suficientemente funcional para que los usuarios finales puedan interactuar con él y proporcionar comentarios.

### 4. Evaluación del Prototipo

En esta etapa, los usuarios finales y otras partes interesadas evalúan el prototipo y proporcionan comentarios y sugerencias para mejorarlo. Se recopila información sobre las deficiencias y se identifican áreas de mejora. Esta retroalimentación es fundamental para el refinamiento continuo del prototipo.

### 5. Mejora Iterativa

En esta etapa, se realizan mejoras incrementales en el prototipo en función de la retroalimentación recibida. Los desarrolladores implementan cambios y ajustes para abordar las deficiencias y agregar nuevas funcionalidades. Este proceso de mejora iterativa continúa hasta que se alcance un prototipo final de alta calidad.

### 6. Implementación Completa

Una vez que se ha alcanzado un prototipo final satisfactorio, se procede a la implementación completa del software. En esta etapa, se desarrolla el código final, se realizan pruebas exhaustivas y se prepara el producto para su lanzamiento al mercado o su implementación en un entorno de producción.

¿Porque decidimos usar esta metodología?:

Se optó por emplear esta metodología debido a nuestra intención de llevar a cabo un desarrollo de software ágil, permitiéndonos realizar modificaciones en el

programa sin ocasionar retrasos en el desarrollo del programa. Además, buscamos mantener una versión funcional del programa que podamos presentar al profesor. Esto nos permitirá recibir retroalimentación oportuna y, a su vez, obtener asistencia para efectuar las correcciones necesarias en nuestro programa. De esta manera, aseguramos que el desarrollo se mantenga alineado con los objetivos del proyecto y las expectativas académicas.

# DIAGRAMA DE FLUJO DE DATOS

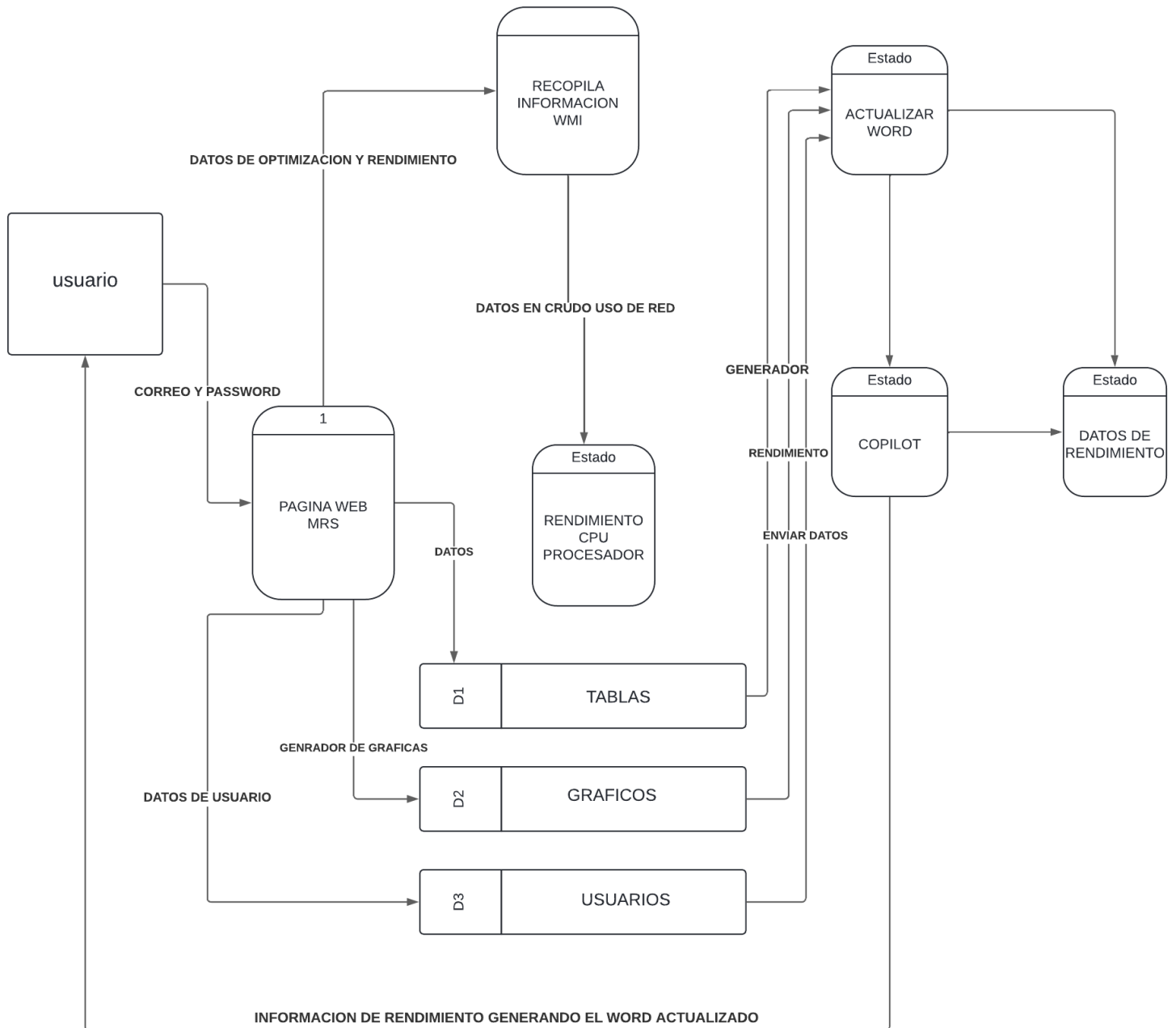
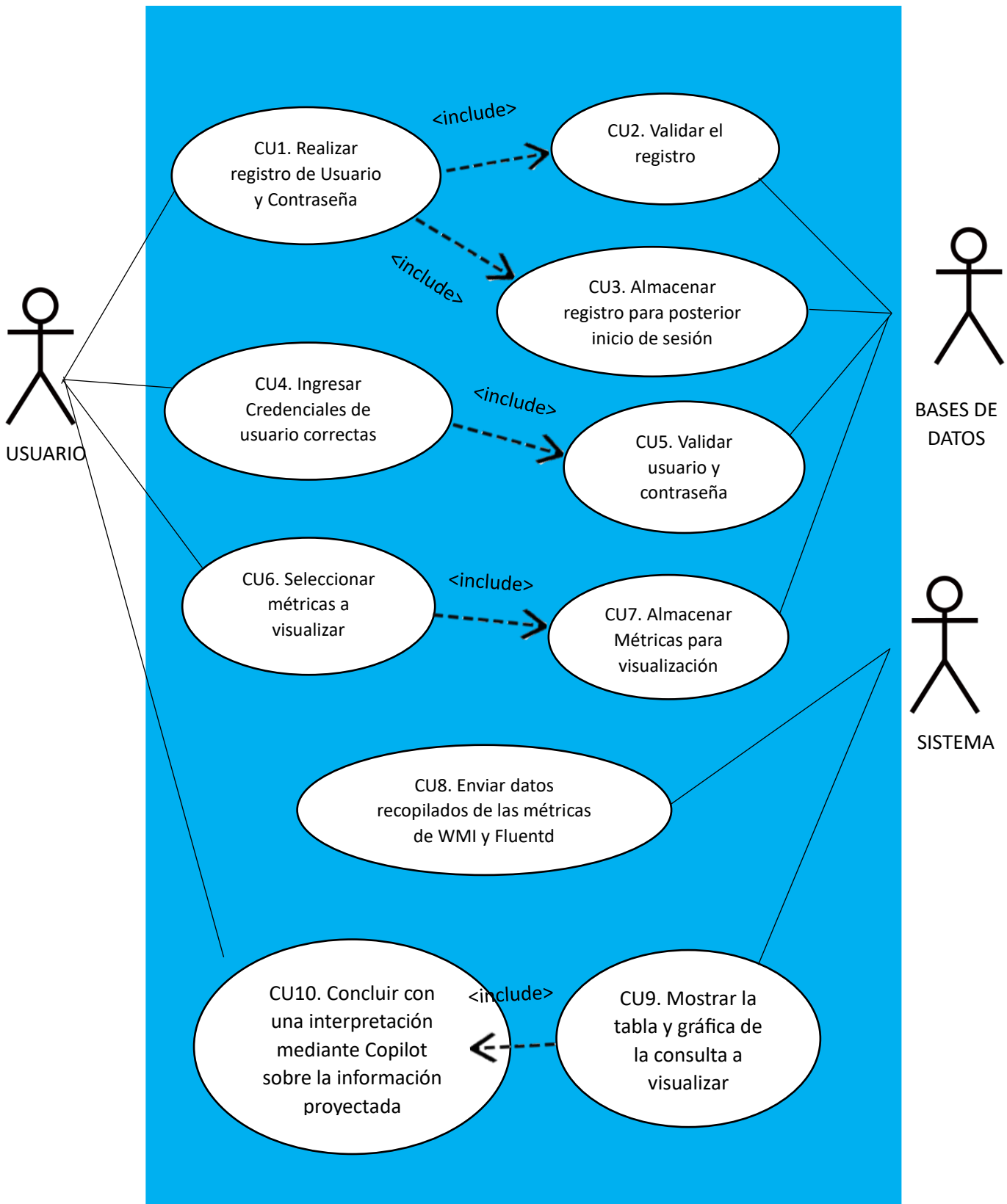


Ilustración 4 DIAGRAMA DE DATOS

# DIAGRAMA DE CASO DE USO

## VISUALIZACIÓN DE RENDIMIENTO



## Diagrama de Flujo Lógico

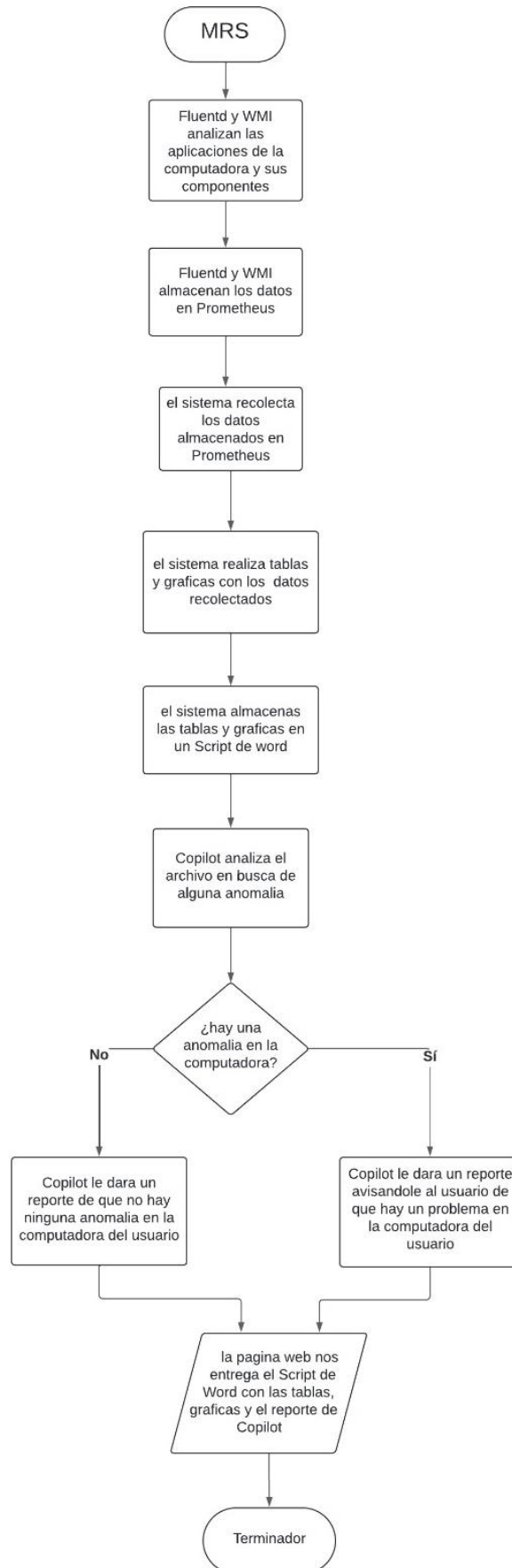


Ilustración 5 Diagrama de Flujo de Datos

# Diccionario de Datos

NOMBRE	TIPO (TIPO DE DATO)	PESO (EN BYTES)	LONGITUD	PESO POR LONGITUD	DESCRIPCION
Metrica	Varchar	2 bytes	30	60 bytes	Nombre de la metrica obtenida del sistema de monitoreo.
ID Nucleo	INT	4 bytes	3	12 bytes	Identificador del nucleo del procesador.
Instancia	Varchar	2 bytes	20	40 bytes	Instancia especifica del sistema o servicio monitoreado.
Tarea	Varchar	2 bytes	20	40 bytes	Nombre de la tarea o trabajo asociado a la métrica.
CPU	Varchar	2 bytes	20	40 bytes	Modo especifico del CPU asociado a la métrica.
Valor	Float	8 bytes	15	120 bytes	Valor numerico asociado a la metrica obtenida del sistema de monitoreo.
				312 bytes	

Ilustración 6 Diccionario de Datos

TOTAL POR REGISTRO	312 bytes
TOTAL POR TABLA	315 bytes
PROYECCION ANUAL	3.12
FECHA DE ACTUALIZACION	Sin actualización

## ESTIMACIÓN ECONÓMICA PROYECTO DE 704 HORAS = 4 MESES

\$18,000 1 Desarrollador / 22 días / 8 horas = \$102 x 704 horas = \$71,808

\$12,000 1 Diseñador / 22 días / 8 horas = \$68 x 704 horas = \$47,872

\$10,000 1 Documentador / 22 días / 8 horas = \$56 x 704 horas = \$39,424

\$30,000 1 Project Manager / 22 días / 8 horas = \$170 x 704 horas = \$119,680

\$71,808 + \$47,872 + \$39,424 + \$119,680 = \$278,784 → Costos de Desarrollo

### Gastos

Luz = \$300 / 22 días / 8 horas = \$1.7 x 704 horas = \$1197

Internet = \$300 / 22 días / 8 horas = \$1.7 x 704 horas = \$1197

Teléfono = \$300 / 22 días / 8 horas = \$1.7 x 704 horas = \$1197

Celular = \$300 / 22 días / 8 horas = \$1.7 x 704 horas = \$1197

Transporte = \$300 / 22 días / 8 horas = \$1.7 x 704 horas = \$1197

Licencia Copilot + Microsoft = \$510 / 22 días / 8 horas = \$2.9 x 704 horas = \$2041

\$1197 + \$1197 + \$1197 + \$1197 + \$1197 + \$2041 = \$8026

### Amortización

Equipo de Cómputo: \$18,000 x .20 (Ley de amortización) = \$3600 / 365 Días / 24 Horas x 704 Horas x 4 (Miembros del Proyecto) = \$1157

Mobiliario: \$5000 x .12 (Ley de amortización) = \$600 / 365 Días / 24 Horas x 704 Horas x 4 (Miembros del Proyecto) = \$193

\$8026 + \$1157 + \$193 = \$9376 → Gastos de Desarrollo

\$278,784

+ \$9376

-----

\$288,160

+30% (GANANCIA) = (\$288,160 x .30) / (1 - .30) = \$123,497

-----

\$411,657

+ 22% (Riesgo) = (\$411,657 x .22) / (1 - .22) = \$116,108

-----

\$527,765

+ 30 % (ISR) = \$527,765 x 1.3 (ISR) = \$686,094

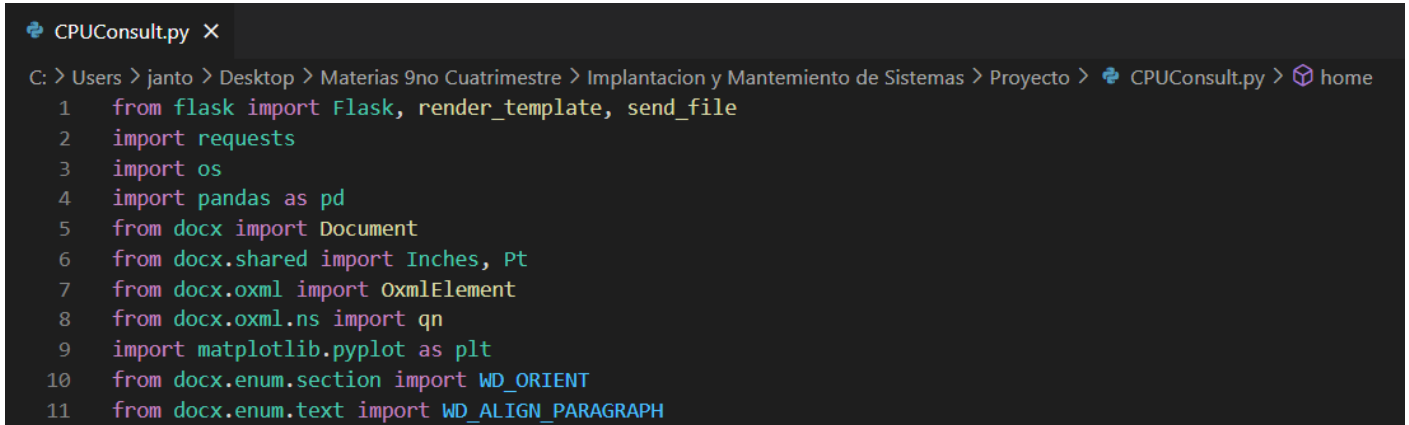
-----

\$686,094

+ 16% (IVA) = \$686,094 x 1.16 (IVA) = \$795,869

**\$795,869 PRECIO FINAL**

# Código



```
1 from flask import Flask, render_template, send_file
2 import requests
3 import os
4 import pandas as pd
5 from docx import Document
6 from docx.shared import Inches, Pt
7 from docx.xml import XmlElement
8 from docx.xml.ns import qn
9 import matplotlib.pyplot as plt
10 from docx.enum.section import WD_ORIENT
11 from docx.enum.text import WD_ALIGN_PARAGRAPH
```

Ilustración 7 código 1

En la primera parte se importan las bibliotecas necesarias para el funcionamiento del script de Python

Flask: Es la biblioteca que va a permitirnos crear la aplicación web y manejar las rutas o APIS que vamos a estar manejando, en este caso, nuestra base de datos que es Prometheus y las APIS que nos arrojan métricas como WMI (Métricas de Hardware) y Fluentd (Métricas de aplicaciones).

requests: Nos permite hacer solicitudes a otras APIs, en este caso, a la API de Prometheus que nos proporciona las métricas del sistema.

os: Nos ayuda a manejar archivos y rutas en nuestra computadora.

pandas: Nos ayuda a organizar y manejar datos en forma de tablas, lo cual es útil para procesar la información obtenida de Prometheus.

docx: Nos permite crear y modificar documentos de Word, donde presentaremos los datos y gráficos.

matplotlib. pyplot: Nos permite crear gráficos que incluiremos en el documento de Word.

docx.shared, docx. xml, docx.enum: Esta es una sub-biblioteca que ayuda a personalizar el formato y la apariencia de los documentos de Word, (el formato de la tabla y la gráfica).



```
13 app = Flask(__name__)
```

*Ilustración 8 código 2*

Creamos una instancia de Flask, en la que en otras palabras es como si dijéramos "quiero crear una nueva aplicación web" y la llamamos "app". Flask es como la estructura de la casa que estamos construyendo.

```
15 @app.route('/')  
16 def home():  
17     return render_template('Pagina.html')
```

*Ilustración 9 código 3*

@app.route('/'): Es donde definimos la ruta principal en este caso ('/') de nuestra aplicación web. Cuando alguien visita la página principal, se ejecutará esta función.

def home ():: Esta función se ejecuta cuando alguien visita la ruta que definimos en @app.route ('/').

return render\_template('Pagina.html'): Indica que se debe mostrar la página web que llamamos "Pagina.html".

```
19 @app.route('/consulta_cpu')  
20 def consulta_cpu():
```

*Ilustración 10 código 4*

@app.route('/consulta\_cpu'): Definimos la ruta '/consulta\_cpu' que se ejecutará cuando alguien la visite.

def consulta\_cpu():: Esta función se ejecuta cuando alguien visita '/consulta\_cpu' igual que con la parte anterior del código.

```

22     url = "http://localhost:9090/api/v1/query"
23
24     params = {
25         "query": "windows_cpu_time_total"
26     }

```

*Ilustración 11 código 5*

url: Es la dirección de la API de Prometheus desde donde obtenemos los datos que almacenamos de WMI en el caso de esta consulta del rendimiento de la CPU.

params: Son los parámetros que enviamos a la API para especificar qué datos queremos (en este caso, windows\_cpu\_time\_total).

```

28     try:
29         response = requests.get(url, params=params)
30         response.raise_for_status()
31

```

*Ilustración 12 código 6*

requests.get(url, params=params): Hacemos una solicitud a la API de Prometheus para obtener los datos.

response.raise\_for\_status(): Verifica si la solicitud fue exitosa. Si hay un error, se genera una excepción en la que no se generara la consulta.

```

32     data = response.json()
33
34     if 'data' in data and 'result' in data['data'] and data['data']['result']:
35         headers = ['Métrica', 'ID Núcleo', 'Instancia', 'Tarea', 'Modo de CPU', 'Valor']
36
37         rows = []
38         for result in data['data']['result']:
39             row = {
40                 'Métrica': result['metric'].get('__name__', 'Desconocido'),
41                 'ID Núcleo': result['metric'].get('core', 'Desconocido'),
42                 'Instancia': result['metric'].get('instance', 'Desconocido'),
43                 'Tarea': result['metric'].get('job', 'Desconocido'),

```

*Ilustración 13 código 7*

```

44         'Modo de CPU': result['metric'].get('mode', 'Desconocido'),
45         'Valor': float(result['value'][1])
46     }
47     rows.append(row)
48
49 df = pd.DataFrame(rows, columns=headers)
50

```

*Ilustración 14 código 8*

`data = response.json()`: Convierte la respuesta de la API en un formato que Python puede entender (un diccionario de Python). Este diccionario nos sirve para organizar y almacenar los datos obtenidos de la API de Prometheus de una manera estructurada y comprensible. Estos diccionarios contienen información sobre el rendimiento de la CPU, como el nombre de la métrica, el núcleo de la CPU, la instancia, la tarea, el modo de la CPU y el valor medido.

`if 'data' in data and 'result' in data['data'] and data['data']['result']`: Verifica si la respuesta contiene los datos esperados (suficientes para hacer la consulta y generar los datos en la tabla y la gráfica).

`headers`: Define los nombres de las columnas para los datos que vamos a procesar como: Métrica, ID Núcleo, Instancia, Tarea, Modo de CPU y Valor.

`rows`: Es una lista donde guardaremos cada fila de datos.

`for result in data['data']['result']`: Recorre cada resultado en los datos obtenidos de la API.

`row = {...}`: Aquí, creamos un diccionario llamado `row`. Cada clave (como 'Métrica', 'ID Núcleo', etc) se asocia con un valor extraído del resultado de la API. Por ejemplo, `result['metric'].get('__name__', 'Desconocido')` obtiene el nombre de la métrica y lo asigna a la clave 'Métrica'. Si la clave no existe, se asigna el valor 'Desconocido' lo que en este caso no sucede, cada dato este asignado a cada clave como ID Núcleo o Modo de CPU.

`rows.append(row)`: Añade cada diccionario a la lista de filas.

`df = pd.DataFrame(rows, columns=headers)`: Convierte la lista de diccionarios en una tabla usando pandas.

```

51 script_dir = os.path.dirname(os.path.abspath(__file__))
52 docx_file_path = os.path.join(script_dir, 'RendimientoCPU.docx')
53 doc = Document()
54
55 section = doc.sections[-1]
56 section.orientation = WD_ORIENT.LANDSCAPE
57 new_width, new_height = section.page_height, section.page_width
58 section.page_width = new_width
59 section.page_height = new_height
60

```

*Ilustración 15 código 9*

`script_dir`: Obtenemos la ubicación del archivo de script para poder guardar el documento de Word en el mismo directorio.

`doc = Document()`: `Document()`: Esta función crea un nuevo documento de Word en blanco usando la biblioteca `python-docx`. En otras palabras, se inicia la creación de un nuevo documento de Word.

1. `os.path.abspath(__file__)`: `__file__` es una variable que contiene la ruta del archivo de script actual. `os.path.abspath(__file__)` convierte esta ruta en una ruta absoluta (completa).
2. `os.path.dirname(...)`: Obtiene el directorio (la carpeta) que contiene el archivo de script. Es como obtener la dirección de tu casa a partir del nombre de la casa.

`docx_file_path`: Define la ruta del archivo de Word que vamos a crear. Se crea una ruta que indique exactamente dónde se guardará el archivo `RendimientoCPU.docx`.

1. `os.path.join(...)`: Combina el directorio del archivo de script (`script_dir`) con el nombre del archivo que queremos crear (`RendimientoCPU.docx`). Esto crea una ruta completa al archivo de Word que vamos a guardar.

`doc.sections[-1]`: Obtiene la última sección del documento. En un nuevo documento, solo hay una sección, así que estamos obteniendo esa sección. Accedemos a la sección del documento para poder modificar sus propiedades, en este caso la orientación de la página de vertical a horizontal.

`section.orientation`: Esta propiedad define la orientación de la página.

`WD_ORIENT.LANDSCAPE`: Cambia la orientación a horizontal.

`section.page_height` y `section.page_width`: Estas propiedades obtienen la altura y el ancho actuales de la página.

`new_width, new_height = section.page_height, section.page_width`: Intercambia los valores de ancho y alto para adaptarlos a la nueva orientación horizontal.

`section.page_width = new_width` y `section.page_height = new_height`: Aplica los nuevos valores de ancho y alto.

```
61 heading = doc.add_heading('Rendimiento de CPU', 0)
62 heading.alignment = WD_ALIGN_PARAGRAPH.CENTER
```

Ilustración 16 código 10

`doc.add_heading('Rendimiento de CPU', 0)`: Añade un título al documento con el texto 'Rendimiento de CPU'. El 0 indica que es un título principal.

`heading.alignment = WD_ALIGN_PARAGRAPH.CENTER`: Centra el título en la página.

```
63 table = doc.add_table(rows=1, cols=len(headers))
64 table.style = 'Table Grid'
```

Ilustración 17 código 11

`doc.add_table(rows=1, cols=len(headers))`: Crea una tabla con una fila y tantas columnas como encabezados hay. `rows=1` significa que comenzamos con una fila para los encabezados.

`table.style = 'Table Grid'`: Aplica un estilo de cuadrícula a la tabla, lo que agrega líneas entre las celdas para hacerlas más visibles.

```
65 hdr_cells = table.rows[0].cells
66 for i, header in enumerate(headers):
67     hdr_cells[i].text = header
68     hdr_cells[i].paragraphs[0].runs[0].bold = True
69     hdr_cells[i].paragraphs[0].runs[0].font.size = Pt(12)
70     hdr_cells[i].paragraphs[0].runs[0].font.name = 'Arial'
```

Ilustración 18 código 12

`hdr_cells = table.rows[0].cells`: Obtiene las celdas de la primera fila de la tabla, que se usarán para los encabezados.

`for i, header in enumerate(headers):`: Recorre los encabezados definidos anteriormente.

hdr\_cells[i].text = header: Establece el texto de cada celda de encabezado.

hdr\_cells[i].paragraphs[0].runs[0].bold = True: Hace el texto en negrita.

hdr\_cells[i].paragraphs[0].runs[0].font.size = Pt(12): Establece el tamaño de la fuente a 12 puntos.

hdr\_cells[i].paragraphs[0].runs[0].font.name = 'Arial': Establece la fuente a Arial.

```
72         for row in rows:
73             row_cells = table.add_row().cells
74             row_cells[0].text = row['Métrica']
75             row_cells[1].text = row['ID Núcleo']
76             row_cells[2].text = row['Instancia']
77             row_cells[3].text = row['Tarea']
78             row_cells[4].text = row['Modo de CPU']
79             row_cells[5].text = str(row['Valor'])
80
81         for cell in row_cells:
82             cell.paragraphs[0].runs[0].font.size = Pt(11)
83             cell.paragraphs[0].runs[0].font.name = 'Arial'
84
```

Ilustración 19 código 13

for row in rows: Recorre cada fila de datos.

row\_cells = table.add\_row().cells: Añade una nueva fila a la tabla y obtiene sus celdas.

row\_cells[0].text = row['Métrica']: Establece el texto de cada celda con los datos correspondientes.

row\_cells[1].text = row['ID Núcleo']: Similar, pero para la siguiente columna.

row\_cells[2].text = row['Instancia']: Y así sucesivamente para todas las columnas.

cell.paragraphs[0].runs[0].font.size = Pt(11): Establece el tamaño de la fuente de los datos en 11 puntos.

cell.paragraphs[0].runs[0].font.name = 'Arial': Establece la fuente de los datos a Arial.

```
86         for row in table.rows:
87             for cell in row.cells:
88                 cell_borders = cell._element.findall('.//w:tcBorders', namespaces={'w': 'http://schemas.openxmlformats.org/wordprocessingml/2006/main'})
89                 if not cell_borders:
90                     tc_pr = cell._element.get_or_add_tcPr()
91                     tc_borders = OxmlElement('w:tcBorders')
92                     for border_name in ['top', 'left', 'bottom', 'right']:
93                         border = OxmlElement(f'w:{border_name}')
94                         border.set(qn('w:val'), 'single')
95                         border.set(qn('w:sz'), '4')
96                         border.set(qn('w:space'), '0')
97                         border.set(qn('w:color'), '000000')
98                         tc_borders.append(border)
99                     tc_pr.append(tc_borders)
```

Ilustración 20 código 14

for row in table.rows:: Recorre cada fila de la tabla.

for cell in row.cells:: Recorre cada celda de la fila.

cell\_borders = cell.\_element.findall('.//w:tcBorders', namespaces={'w': 'http://schemas.openxmlformats.org/wordprocessingml/2006/main'}): Busca si ya existen bordes en la celda.

if not cell\_borders:: Si no existen bordes, crea y añade nuevos bordes.

tc\_pr = cell.\_element.get\_or\_add\_tcPr(): Obtiene o crea propiedades para la celda.

tc\_borders = XmlElement('w:tcBorders'): Crea un nuevo elemento de bordes.

for border\_name in ['top', 'left', 'bottom', 'right']:: Añade bordes a todos los lados de la celda.

border = XmlElement(f'w:{border\_name}'): Crea un borde para un lado específico.

border.set(qn('w:val'), 'single'): Define el tipo de borde como simple.

border.set(qn('w:sz'), '4'): Define el tamaño del borde.

border.set(qn('w:space'), '0'): Define el espacio del borde.

border.set(qn('w:color'), '000000'): Define el color del borde como negro.

tc\_borders.append(border): Añade el borde al elemento de bordes.

tc\_pr.append(tc\_borders): Añade el elemento de bordes a las propiedades de la celda.

```
101 plt.figure(figsize=(10, 5))
102 df.groupby('Modo de CPU')['Valor'].mean().plot(kind='line', marker='o')
103 plt.xlabel('Modo de CPU')
104 plt.ylabel('Valor de CPU')
105 plt.title('Valores de CPU por Modo')
106 plt.tight_layout()
107
108 line_chart_path = os.path.join(script_dir, 'cpu_line_chart.png')
109 plt.savefig(line_chart_path)
110 plt.close()
```

Ilustración 21 código 15

plt.figure(figsize=(10, 5)): Crea una nueva figura para la gráfica con tamaño 10x5 pulgadas.

df.groupby('Modo de CPU')['Valor'].mean().plot(kind='line', marker='o'): Agrupa los datos por el modo de CPU y calcula el valor promedio, luego crea una gráfica de líneas con marcadores.

plt.xlabel('Modo de CPU') y plt.ylabel('Valor de CPU'): Establece etiquetas para los ejes "x" y "y" de la gráfica.

plt.title('Valores de CPU por Modo'): Establece un título para la gráfica.

plt.tight\_layout(): Ajusta automáticamente el diseño de la gráfica para que se vea bien.

```
112 doc.add_heading('Gráfica de Líneas: Valores de CPU por Modo', level=1)
113 doc.add_picture(line_chart_path, width=Inches(6))
```

Ilustración 22 código 16

doc.add\_heading('Gráfica de Líneas: Valores de CPU por Modo', level=1): Añade un título al documento de Word para la gráfica.

doc.add\_picture(line\_chart\_path, width=Inches(6)): Inserta la imagen de la gráfica en el documento de Word. width=Inches(6) indica que la imagen se ajustará a un ancho de 6 pulgadas en el documento.

```
115 doc.save(docx_file_path)
116 return send_file(docx_file_path, as_attachment=True, download_name='RendimientoCPU.docx')
117
118 else:
119     return "No se encontraron resultados para la consulta."
120
121 except requests.exceptions.RequestException as e:
122     return f"Error al realizar la solicitud HTTP: {e}"
123
124 if __name__ == '__main__':
125     app.run(debug=True)
```

Ilustración 23 código 17

doc.save(docx\_file\_path): Guardamos el documento de Word completo que hemos construido hasta ahora. docx\_file\_path es la ruta completa al archivo donde se guardará el documento. Este archivo ahora contiene el título, la tabla con los datos de rendimiento de la CPU y la gráfica de líneas.

Return send\_file(docx\_file\_path, as\_attachment=True, download\_name='RendimientoCPU.docx'): Después de guardar el documento, esta



línea devuelve el archivo como una respuesta HTTP para que el usuario lo pueda descargar.

else:: Si no se encontraron resultados para la consulta a la API de Prometheus, el código retorna un mensaje simple: “No se encontraron resultados para la consulta”.

Manejo de Excepciones (except requests. exceptions. RequestException as e): Esta parte captura cualquier error que pueda ocurrir durante la solicitud HTTP a la API de Prometheus. Si ocurre un error, se devuelve un mensaje que describe el problema: “Error al realizar la solicitud HTTP”

```
if __name__ == '__main__':
```

```
    app.run(debug=True):
```

Finalmente, la parte final del código asegura que si ejecutas este script directamente desde la línea de comandos con `python nombre_del_script.py`, Flask iniciará su servidor web integrado y comenzará a escuchar las solicitudes HTTP entrantes en el puerto predeterminado (generalmente el puerto 5000).

Si este script se importa como un módulo en otro script, `__name__` no será igual a `'__main__'`, y por lo tanto, `app.run(debug=True)` no se ejecutará automáticamente, lo cual es deseable para evitar iniciar múltiples servidores si este script se usa en un entorno de importación.

## Alternativas

Al implementar una Plataforma de Monitoreo de Rendimiento de Software, debemos de considerar que entramos a un mercado de gran competencia, en el que debemos tener en cuenta todas las posibles alternativas a nuestro Software. Hay que distinguarnos de la competencia y por supuesto también nos corresponde destacarnos de entre los servicios que ellos ofrezcan al mercado como alternativa a los nuestro, por ello presentamos algunas plataformas que comparten algo en común con nuestro software y también el como nosotros ofrecemos un factor de cambio mayor a la de nuestra competencia.

New Relic es una plataforma de monitoreo de rendimiento de aplicaciones que proporciona una visión completa del rendimiento de aplicaciones e infraestructura. Permite a los equipos de desarrollo y operaciones obtener métricas detalladas, trazas de transacciones y monitoreo en tiempo real.

Trazas distribuidas: Permite rastrear solicitudes a través de múltiples servicios para identificar cuellos de botella y problemas de rendimiento.

Alertas: Configuración de alertas basadas en umbrales específicos para recibir notificaciones cuando el rendimiento cae por debajo de los niveles aceptables.

Análisis de errores: Ofrece detalles sobre errores y excepciones en el código para facilitar la resolución de problemas.

Dashboards personalizables: Los usuarios pueden crear paneles personalizados para visualizar las métricas más relevantes para su contexto.

Datadog es una plataforma de monitoreo y análisis que se integra con una amplia gama de servicios para proporcionar una vista unificada de la infraestructura, los registros y las trazas de aplicaciones. Es conocida por su capacidad de monitoreo en tiempo real y su flexibilidad.

Monitoreo en tiempo real: Proporciona datos en tiempo real sobre el rendimiento de las aplicaciones y la infraestructura.

Alertas configurables: Permite establecer alertas basadas en condiciones específicas y umbrales definidos por el usuario.

**Dashboards personalizables:** Los usuarios pueden crear paneles personalizados para monitorear las métricas que consideren más importantes.

**Integración con múltiples servicios:** Ofrece integraciones con una amplia gama de herramientas y servicios, facilitando la consolidación de datos de diferentes fuentes.

**AppDynamics** es una solución de monitoreo de rendimiento que proporciona una visibilidad completa del rendimiento de las aplicaciones y la infraestructura subyacente. Está diseñada para ayudar a los equipos a identificar y resolver problemas de rendimiento rápidamente.

**Monitoreo de usuario final:** Ofrece visibilidad sobre la experiencia del usuario final, lo que ayuda a identificar problemas que afectan directamente a los usuarios.

**Análisis de rendimiento de aplicaciones:** Proporciona análisis detallados del rendimiento de las aplicaciones para detectar cuellos de botella y optimizar el rendimiento.

**Diagnóstico de problemas:** Herramientas para identificar y diagnosticar problemas de rendimiento, facilitando la resolución rápida.

**Integración con herramientas DevOps:** Se integra con diversas herramientas de DevOps, permitiendo un flujo de trabajo más eficiente y colaborativo.

## Componentes Clave de nuestra Plataforma

Ya definimos y estudiamos a nuestra competencia, pero ¿Cómo nos destacamos nosotros de ellos? ¿Que ofrecemos que ellos no? ¿Y es suficiente factor para ser elegidos de entre todos ellos? Bueno, nosotros trazamos nuestros objetivos en el protocolo de este mismo proyecto anteriormente adjunto, nosotros queremos ofrecer lo siguiente:

Las métricas son esenciales para nosotros porque por medio de estos valores numéricos podremos representar el estado y el rendimiento de diversos aspectos de cualquier aplicación o infraestructura de cualquier empresa, por eso mediante una base de datos de series temporales, podremos monitorear las métricas de cada cosa en tiempo real.

Las métricas son esos datos numéricos que, al ser recopilados a intervalos regulares desde diversas fuentes, como aplicaciones, servicios, y componentes de infraestructura nos permitirán conocer el rendimiento, la disponibilidad y la escalabilidad de cualquier aplicación o infraestructura que estaremos diagnosticando.

Las trazas distribuidas permiten rastrear el recorrido de las solicitudes a través de los diferentes servicios que componen cualquier aplicación, ayudando a identificar cuellos de botella y problemas de rendimiento.

Los registros (logs) contienen detalles sobre los eventos que ocurren dentro de la aplicación, lo que es crucial para la depuración y el análisis post-mortem.

Fluentd: Es un agregador de registros de código abierto que unifica la recolección y el consumo de datos de registros con el recolectaremos y enviaremos registros a sistemas como en nuestro caso Prometheus, sistemas de archivos, bases de datos, etc.

# Manual de Usuario o Guion de pruebas

## Monitoreo de Rendimiento de Software (MRS)

MRS es un programa para monitorear, medir y evaluar el rendimiento de la computadora del usuario. El programa mediante la integración de otros programas como Prometheus, Grafana, Fluentd y Alertmanager escaneará la computadora del usuario tanto el hardware y el software para posteriormente hacer gráficas y tablas del estado actual de la computadora las cuales Copilot se encargará de analizar para ver si se debe enviar o no una alerta al usuario para que intervenga y solucione la anomalía.

### Pasos para usar el programa MRS

Consultar CPU:

*Paso 1:* el usuario dará clic en el icono del programa para que el programa se ejecute.

*Paso 2:* cuando el programa se abra el usuario dará clic en el enlace que hay en el programa el cual lo enviara a la página web del programa.

*Paso 3:* el usuario dará clic en el botón realizar “consultar CPU” lo que va a descargar un archivo de Word donde estará plasmada las métricas en una tabla y en grafica para que el usuario lo pueda consultar y analizar

*Paso 4:* Word utilizando la herramienta de Copilot analizara el archivo de Word para que los usuarios sin mucho conocimiento puedan entender los datos

## Retos y Esfuerzos

Erick Jovanny Ramírez Aguilar: uno de los retos que tuve a la hora de realizar este proyecto fue la organización, debido a nuestras otras materias y a situaciones externar la organización del equipo no fue la mejor ya que todos teníamos otros trabajos que realizar y cuando ya teníamos un poco de tiempo sucedió algún imprevisto por lo cual no pudimos realizar el trabajo con una mejor organización, aunque si pudimos cumplir con los tiempos. Otro reto que tuve fue el usar el lenguaje Python debido a mi desconocimiento ya que Python era un lenguaje nuevo para mí por lo cual era difícil de entender y tenía que pedirles ayuda a mis compañeros para poder entender los códigos

José Antonio Sánchez Cerda: Fue difícil organizarnos en equipo por como distribuiríamos el trabajo ya que solo 1 podía hacer uso de Copilot, y además necesitamos configurar nuestra Base de Datos (Prometheus) así como las APIs como WMI en cada una de nuestras computadoras con tal de usar de forma más efectiva cada uno de nuestros programas, por lo que el configurar todo esto en cada uno de nuestros equipos hubiera sido no solo un reto, si no una inversión de tiempo que no hubiera sido del todo productiva o provechosa dado el poco tiempo que teníamos con respecto a cada avance, finalmente tuvimos que aportar todos ya fuera por la parte lógica del proyecto (Todo el diseño y el código del Proyecto) así como la parte de la documentación. Otro reto fue el hecho de que no estábamos del todo familiarizados con cómo funcionaba Copilot, por lo que en múltiples ocasiones tuvimos que cambiar el curso de nuestro proyecto, invirtiendo tiempo de más en hallar un concepto más viable y útil de lo que teníamos en mente al principio al establecer nuestras metas y objetivos. Finalmente pudimos lograr una versión estable de nuestro proyecto, y ya hemos planteado como funcionará este mismo una vez teniendo este Prototipo como base

Erick Salvador Veloz González: El desarrollo de un programa de monitoreo de rendimiento de software presentó ciertos desafíos que se debían superar. Uno de los principales retos fue el diseño de una página web visualmente atractiva y

funcional que se conectara con la aplicación principal, encargada del monitoreo mediante Prometheus y Copilot. Crear una interfaz intuitiva y accesible para que los usuarios pudieran interactuar con las métricas de rendimiento fue un proceso complejo. La integración con las herramientas de monitoreo debía ser perfecta, garantizando que los datos se mostraran en tiempo real sin latencias significativas.

La página web también debía ser responsiva, permitiendo un uso eficiente en diferentes dispositivos y tamaños de pantalla.

En la parte organizacional los retos más significativos fue la disponibilidad limitada de Copilot, ya que solo un equipo de cómputo contaba con su uso. Esto afectó la colaboración y la productividad del equipo, La coordinación y la comunicación efectiva entre los miembros del equipo fueron esenciales para superar estos obstáculos.

A pesar de las dificultades, el proyecto avanzó gracias a la planificación, la colaboración y el uso adecuado de las herramientas disponibles, resultando en una solución robusta y eficiente para el monitoreo de rendimiento de software.

Alexis David Ceballos Cadena:

Reto: Fue adaptarme con cada uno de los integrantes con los del equipo que me toco interactuar que fueron mis compañeros en este proyecto. Antonio Erick y Jovanny fue un reto por que hicieron sacar lo mejor de mi

Y trabajar con compañeros realmente comprometidos no había trabajado con personas en mi carrera y eso es que este proyecto sea uno de los mejores gracias a cada uno de los integrantes de este equipo

Para mi y mi equipo fue investigar y aprender mas sobre el funcionamiento de Copilot y las herramientas de optimización como fluentd y wmi Lo que más me costó trabajo fue la documentación ya que gracias al apoyo de mis compañeros pudimos realizarla a tiempo sobre este gran proyecto de MRS.

## Conclusiones

Erick Jovanny Ramírez Aguilar: este proyecto ha sido el más desafiante hasta el momento debido a uso de la herramienta de Copilot ya que es una herramienta que tiene muy poco tiempo de ser creada además de por lo cual no sabíamos cómo funcionaba exactamente y no se nos ocurría una idea de cómo implementarlo en nuestro programa ya que Copilot era que un requisito que tenía que tener nuestro programa para este proyecto. También durante este proyecto tuvimos que volver a usar nuestros conocimientos para realizar los diagramas como el de caso de uso y los diagramas de flujo los cuales se me hicieron un poco pesado realizarlos ya que no recordaba cómo se realizaban.

José Antonio Sánchez Cerda: Fue un proyecto que en lo personal me costó algo de trabajo porque nuevamente no tenía una idea muy clara de cómo podríamos integrar Copilot ya que nunca había hecho uso de él, solamente había escuchado que era una inteligencia artificial y nada más que eso. Tuve que desarrollar una idea que fuera viable para que el proyecto finalmente cobrará vida y sentido de utilidad para un entorno de la vida real como es en el ámbito de las empresas, por lo que teniendo como base nuestro primer Prototipo funcional pudimos empezar a desenvolver más y más ideas que solo iban a mejorar mucho más nuestro aplicativo de lo que ya estaba siendo. El trabajo en equipo por supuesto que fue difícil porque hubo un punto en el que me llegue a desenvolver más como un jefe más que como un líder, sin embargo, con la ayuda de los muchachos y la mía, como equipo pudimos retomar rumbo y finalmente logramos lo que nos habíamos propuesto desde un principio, obteniendo finalmente empezar a trabajar más de lleno en el proyecto y disfrutando del proceso.

Erick Salvador Veloz González: Desde mi perspectiva, el desarrollo de este programa de monitoreo de rendimiento de software fue una experiencia desafiante. Crear la página web, que se convirtió en la pieza central de interacción para los usuarios, fue especialmente interesante para mí. A pesar de los numerosos retos técnicos y organizacionales, como la integración con Prometheus y Copilot y la



limitación de acceso a herramientas clave, el proceso me permitió aplicar y ampliar mis habilidades en diseño y desarrollo web. El resultado final, una interfaz visualmente atractiva y funcional, Este proyecto no solo mejoró mis competencias técnicas, sino que también la importancia de la planificación y la comunicación efectiva en el éxito de cualquier iniciativa tecnológica.

Alexis David Ceballos Cadena: Este fue un proyecto de rendimiento de software tiene como objetivo principal la automatización de empresas mediante el uso de Copilot 365 y las herramientas proporcionadas por Microsoft. La implementación de estas tecnologías tiene el potencial de mejorar significativamente la rapidez y el rendimiento de los usuarios, facilitando la ejecución de tareas de manera más eficiente a través de la inteligencia artificial.

Copilot 365, en particular, ofrece funcionalidades avanzadas que permiten a los usuarios completar tareas y obtener información de manera rápida y precisa, eliminando incertidumbres y reduciendo el tiempo necesario para optimizar el rendimiento del software. Estas mejoras no solo incrementan la productividad, sino que también promueven un uso más efectivo y racionalizado de los recursos tecnológicos disponibles, contribuyendo así al éxito y la competitividad de las empresas en el mercado actual.

# BIBLIOGRAFIA

- (s.f.). Obtenido de <https://www.soluciones-im.com/es/la-monitorizacion-de-sistemas-informaticos-en-8-pasos>
- ¿Qué es PHP? (2001-2024). Obtenido de <https://www.php.net/manual/es/intro-what-is.php>
- Blasi, M. (s.f.). *LaDiferencia.net*. Recuperado el 29 de 07 de 2024, de LaDiferencia.net: <https://www.ladiferencia.net/diferencias-entre-monitoreo-activo-y-monitoreo-pasivo/>
- camillepack, c.-M.-M. (15 de 07 de 2024). <https://learn.microsoft.com/es-es/copilot/microsoft-365/microsoft-365-copilot-overview>. Obtenido de Información general de Microsoft Copilot para Microsoft 365: <https://learn.microsoft.com/es-es/copilot/microsoft-365/microsoft-365-copilot-overview>
- Desconocido. (16 de 7 de 2020). *Bantu group*. Obtenido de Bantu group: <https://www.bantugroup.com/blog/estrategias-para-implementar-procesos-de-mejora-continua#:~:text=Realizando%20un%20feedback%20o%20an%C3%A1lisis%20interno%20de%20cada,dando%20resultados%20positivos%20y%20descartar%20las%20que%20no>
- Desconocido. (30 de 8 de 2023). *AppMaster*. Obtenido de AppMaster: <https://appmaster.io/es/blog/arquitectura-de-software-de-escalabilidad-y-rendimiento>
- desconocido. (27 de 09 de 2023). *Sentrio*. Obtenido de Sentrio: <https://sentrio.io/blog/herramientas-esenciales-de-monitoreo-y-observabilidad/>
- Desconocido. (2 de 7 de 2023). *TecnologiaDigital*. Obtenido de TecnologiaDigital: [https://informatecdigital.com/desarrollo/metodologia-de-prototipo-para-el-desarrollo-de-software/#google\\_vignette](https://informatecdigital.com/desarrollo/metodologia-de-prototipo-para-el-desarrollo-de-software/#google_vignette)
- F.G, C. (s.f.). *Estrategem*. Recuperado el 31 de 07 de 2024, de Estrategem: <https://estrategem.com/blog/escalabilidad/>
- Fernandez, O. (14 de 07 de 2024). *Aprender Big Data*. Obtenido de Aprender Big Data: <https://aprenderbigdata.com/prometheus/>
- Fernandez, Y. (9 de 3 de 2024). *Xataka* . Obtenido de Xataka: <https://www.xataka.com/basics/microsoft-copilot-que-como-funciona-este-chat-inteligencia-artificial>
- INEGI. (30 de noviembre de 2017 a 2020 ). <https://www.inegi.org.mx/programas/endutih/2023/>. Obtenido de Encuesta Nacional sobre Disponibilidad y Uso de Tecnologías de la Información en los Hogares (ENDUTIH) 2023: <https://www.inegi.org.mx/programas/endutih/2023/>
- Informacion General de Microsoft Copilot para Microsoft 365*. (15 de 7 de 2024). Obtenido de <https://learn.microsoft.com/es-es/copilot/microsoft-365/microsoft-365-copilot-overview>
- Información general de Microsoft Copilot para Microsoft 365*. (15 de 7 de 2024). Obtenido de <https://learn.microsoft.com/es-es/copilot/microsoft-365/microsoft-365-copilot-overview>
- Martinez, J. (27 de 12 de 2021). *OpenWebinars*. Obtenido de OpenWebinars: <https://openwebinars.net/blog/que-es-grafana-y-primeros-pasos/>
- Monitoreo de WMI (Windows Management Instrumentation) con OpManager*. (s.f.). Obtenido de <https://www.manageengine.com/latam/network-monitoring/monitoreo-de-wmi.html>
- Sancho, P. (2023 de 04 de 10). *hiberus blog*. Obtenido de hiberus blog: <https://www.hiberus.com/crecemos-contigo/copilot-de-microsoft-que-es-y-como-mejora-tu-productividad/#:~:text=Algunas%20de%20las%20principales%20ventajas%20de%20utilizar%20esta,de%20c%C3%B3digo.%20...%207%20Integraci%C3%B3n%20con%20Teams.%20>
- Sonego, F. (28 de 3 de 2024). *yo quiero programas*. Obtenido de yo quiero programas: <https://yoquieroprogramar.conosur.tech/metricas-de-para-medir-el-rendimiento-de-nuestras-aplicaciones/>
- Wrobel, M. (16 de 6 de 2023). *Invgate*. Obtenido de Invgate: <https://blog.invgate.com/es/despliegue-de-software>

**GRACIAS**