clave i | Software solutions for business

Gestión y optimización del desempeño del sofware empresarial con el Framework SWIM

(Software Workflow Inspection Model)

Hector Gomis
Head of Products Clavei.
hgomis@clavei.es

Francisco Moya
Product & Technolog
fmoya@clavei.es

GOCLOUD

Financiado por la Unión Europea
NextGenerationEU

GOBIERNO DE ESPAÑA | MINISTERIO DE TRANSFORMACIÓN DIGITAL | SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL

red.es

Plan de Recuperación, Transformación y Resiliencia

RESÚMEN: La evaluación del desempeño del software es crucial para garantizar su calidad y eficacia. Cuando además nos enfrentamos a un proceso de migración de aplicaciones con tecnologías legacy desde entornos escritorio hacia otros cloud native, la tarea se convierte en crítica para conseguir que el impacto en los usuarios finales no suponga un freno en los procesos de adopción de los nuevos entornos. En este artículo, se propone el framework (SWIM) y un conjunto de métricas integral para medir diferentes aspectos del desempeño del software, incluyendo rendimiento, fiabilidad, escalabilidad y usabilidad. Se presenta dicho framework de mejora continua y las métricas propuestas, su aplicación práctica y su utilidad en el proceso de desarrollo de software.

PALABRAS CLAVE: framework, legacy, cloud native, migración, cloud computing, sistemas heredados.

ABSTRACT: Evaluating software performance is crucial to ensure its quality and effectiveness. When, in addition, we face a process of migrating applications with legacy technologies from desktop environments to other cloud native environments, the task takes on an essential criticality so that the impact on end users does not represent a brake on the adoption processes towards new environments. In this article, a comprehensive framework (SWIM) and set of metrics will be proposed to measure different aspects of software performance, including performance, reliability, scalability, and usability. This continuous improvement framework and the proposed metrics, their practical application and their usefulness in the software development process will be presented.

KEYWORDS: framework, legacy, cloud native, migration, cloud computing, legacy systems

1. INTRODUCCIÓN

La evaluación del desempeño del software es esencial para garantizar que este cumpla con los requisitos de calidad y funcionalidad. Sin embargo, medir el desempeño de manera efectiva puede ser un desafío debido a la complejidad y la diversidad de los sistemas software. En este artículo, presentamos un framework y un conjunto integral de métricas diseñadas para medir diferentes aspectos del desempeño del software, proporcionando así una visión completa de su calidad y eficacia.

Este artículo se realiza en el marco del proyecto Sistema y metodología de migración de aplicaciones heredadas de gestión empresarial a la Nube. (GOCLOUD), un proyecto de desarrollo experimental en el cual la empresa CLAVEi se encuentra investigando y desarrollando un sistema, metodología y herramientas que permitirán la migración, sin interrupción del servicio, de aplicaciones y bases de datos de paquetes de gestión empresarial multisectoriales desde un paradigma de funcionamiento local y centralizado hacia un paradigma de funcionamiento en la nube. Este proyecto ha sido apoyado por el Ministerio de la Transformación Digital y de la Función Pública, a través del ente Red.es.

La evaluación del desempeño del software ha sido objeto de investigación durante décadas, y se han propuesto numerosas métricas para medir diversos aspectos de su funcionamiento. En la literatura existente, se encuentran métricas que se centran en aspectos específicos del desempeño, como el rendimiento, la fiabilidad, la escalabilidad y la usabilidad, si bien es importante reconocer la necesidad de un enfoque integral que abarque múltiples dimensiones junto con el correcto análisis y seguimiento proporcionado por una metodología integral aplicada sobre estas métricas para que los equipos de producto puedan obtener una imagen completa y detallada de su funcionamiento, permitiéndoles identificar áreas de mejora y tomar medidas correctivas para optimizar su desempeño y calidad.

2. LA APROXIMACIÓN DE LA METODOLOGÍA SWIM

La metodología SWIM (Software Workflow Inspection Model) es un enfoque sistemático diseñado para medir y monitorizar diversos aspectos del desempeño de un software de manera integral. Se centra en recopilar datos relevantes, analizar tendencias y poner en marcha acciones correctivas con el fin de mejorar continuamente el rendimiento del software. A continuación, se presenta un resumen de los pasos principales de la metodología SWIM

1. Definición de objetivos y métricas:
Identificar los objetivos clave del software en términos de calidad, rendimiento, seguridad, usabilidad, etc. Seleccionar métricas específicas que puedan medir el progreso hacia cada objetivo definido.

2. Implementación de herramientas de monitorización:
Seleccionar y configurar herramientas adecuadas para la recopilación automatizada de datos relacionados con las métricas definidas. Integrar estas herramientas en el ciclo de desarrollo y despliegue del software para garantizar la recopilación continua de datos.

3. Recopilación y almacenamiento de datos:
Establecer un sistema centralizado para recopilar y almacenar datos de métricas de desempeño de manera regular y sistemática. Asegurar la integridad y seguridad de los datos recopilados, utilizando prácticas de gestión de datos adecuadas.

4. Análisis y visualización de datos:
Analizar los datos recopilados para identificar tendencias, patrones y áreas de mejora en el desempeño del software. Utilizar técnicas de visualización de datos para representar claramente los resultados del análisis y facilitar la comprensión.

5. Identificación de tendencias y problemas:
Monitorizar regularmente las métricas de desempeño para detectar desviaciones significativas que puedan indicar problemas potenciales. Identificar y priorizar los problemas y áreas de mejora en función de su impacto en los objetivos del software.

6. Establecimiento de acciones correctivas:
Desarrollar planes de acción específicos para abordar los problemas identificados y mejorar el desempeño del software. Implementar medidas correctivas de manera oportuna y efectiva, asignando responsabilidades claras a los miembros del equipo.

7. Evaluación de impacto y retroalimentación:
Evaluar el impacto de las acciones correctivas implementadas mediante la comparación de métricas antes y después de la intervención. Obtener retroalimentación de los usuarios y otras partes interesadas para evaluar la efectividad de las medidas correctivas y la satisfacción del cliente.

8. Iteración y mejora continua:
Revisar y ajustar la metodología SWIM de manera regular en función de la retroalimentación recibida y los cambios en el entorno del software. Buscar oportunidades para optimizar el proceso de medición y seguimiento de métricas de desempeño y mejorar continuamente el rendimiento del software.

El objetivo de la metodología SWIM es proporcionar un marco estructurado y sistemático para gestionar el desempeño del software, garantizando que se cumplan los objetivos del proyecto y se satisfagan las necesidades de los usuarios de manera efectiva.
GB hasta TB en algunos casos. Un reto añadido es fusionar tablas situadas en distintas bases de datos en unas sola, esto implica tener especial cuidado en el mapeo de datos.

3. FASES DE LA METODOLOGÍA SWIM

1. Definición de objetivos y métricas

En esta primera fase, es importante profundizar en la identificación de los objetivos específicos de la migración y en la definición de métricas que utilizaremos para medir el éxito de cada objetivo definido.

a) Análisis de la situación inicial. Se debe evaluar el estado actual del sistema legacy y sus limitaciones, al mismo tiempo que se identifican claramente los principales problemas que se espera resolver con la migración a cloud native.

b) Establecer objetivos SMART. Es importante que, a la hora de determinar los objetivos a conseguir con la migración, estos sean específicos, medibles, alcanzables, relevantes y con un tiempo definido de consecución. Así será más fácil maximizar las posibilidades de éxito y enfocarnos en conseguir los resultados deseados.

c) Priorizar objetivos. Una vez establecidos los objetivos a alcanzar, es preciso establecer una prioridad de consecución en función de su importancia para el éxito en el proceso de adopción de la solución. A la hora de establecer esa priorización, tendremos en cuenta aspectos importantes como los recursos disponibles tanto técnicos como humanos, el plazo máximo establecido para la migración o factores externos que puedan afectar al proceso.

d) Definición de métricas clave. Se identificarán métricas específicas para la medición del progreso hacia la consecución de cada objetivo. Para que sean relevantes y nos ayuden a saber el grado de consecución, estas métricas han de ser relevantes, cuantificables y totalmente alineadas con esos objetivos.

e) Determinar baselines. La determinación de puntos de referencia para poder medir el progreso de cada métrica es fundamental y asegurarnos que el proyecto de migración avanza hacia los objetivos marcados de forma satisfactoria.

2. Implementación de herramientas de monitorización.

La monitorización continua de todo el proceso es fundamental, por lo que será preciso seleccionar y configurar diferentes herramientas que nos permitan monitorizar y evaluar la consecución de los objetivos fijados.

a) Selección de herramientas. Antes de proceder a la selección de una o varias herramientas de monitorización, es preciso realizar una evaluación de las necesidades y requisitos específicos del proyecto, incluyendo el tipo de aplicaciones y servicios que se monitorizarán, los sistemas operativos o las plataformas utilizadas. A la hora de seleccionar estas herramientas hay que valorar de forma minuciosa tanto la facilidad de implementación, configuración y mantenimiento como su capacidad de escalar a medida que el proyecto lo necesite.

b) Configuración de herramientas y testeo. Una vez seleccionadas las herramientas es preciso configurarlas correctamente para asegurarnos de que los datos recopilados son consistentes. Será de vital importancia que estas herramientas estén integradas dentro del propio producto, para que de forma totalmente transparente estén alineadas con el roadmap del producto objeto de análisis y su ciclo de desarrollo. Es importante además la configuración de alertas para detectar condiciones anormales o problemas potenciales. Finalmente, se realizarán pruebas exhaustivas para asegurarnos de que las herramientas están funcionando según lo esperado, recopilando y mostrando los datos de manera fiable.

3. Recopilación y almacenamiento de datos.

En el punto siguiente de esta metodología se realizará un análisis de los datos, pero para que este sea óptimo, es imprescindible que estos se recopilen y almacenen de manera adecuada.

a) Procesos de recopilación de datos. Será preciso definir procesos estructurados y con una documentación clara para la recopilación de datos de monitorización del rendimiento del sistema y la infraestructura cloud. En esos procesos se incluye la configuración de agentes de monitorización, el uso de aplicaciones que faciliten esta recopilación, la implementación de APIs para la extracción de datos o la integración por medio de herramientas externas.
b) Repositorio centralizado. Los datos recopilados, se deberán almacenar en un repositorio centralizado que sea confiable y sea escalable en función del volumen de datos obtenidos durante el proceso.

c) Estructura del almacenamiento de datos. Además de que la estructura de datos que se determine ha de ser clara, coherente y organizada, es imprescindible que sea flexible y pueda adaptarse a cambios en los requisitos del proyecto o en el tipo de da-

tos que se necesita recopilar.

d) Implementación de medidas de seguridad y privacidad. Se deben incluir todas las medidas de seguridad necesarias para asegurarse de la imposibilidad de accesos no autorizados, pérdida, corrupción o encriptación de datos. Ese almacenamiento de datos ha de cumplir con lo establecido en materia de privacidad por la normativa vigente y toda la información debe estar respaldada por copias de seguridad regulares, almacenadas de forma segura, accesible en caso necesario, así como determinar una estrategia de enmascaramiento o anonimización de ciertos datos si fuera necesaria debido a la relevancia de los mismos.

e) Validación y testeo. Previo a la puesta en producción, se deberán realizar pruebas de integración con las herramientas de monitorización, así como de recuperación de datos para garantizar que, en caso de fallo en el sistema o pérdida de datos, será posible recuperarlos en el menor tiempo posible.

4. Análisis y visualización de datos.

La siguiente fase de la metodología nos va a permitir entender el rendimiento del sistema y la infraestructura de forma que podamos identificar problemas, patrones, anomalías o áreas de mejora.

a) Visualización de datos. Una vez que los datos han sido procesados por las diferentes herramientas, es fundamental que estas nos permitan una visualización clara, comprensible y accesible por medio de la representación gráfica de los mismos.

b) Acceso a conjuntos de datos enriquecidos. Definiremos y documentaremos además conjuntos de datos autocontenidos, en forma de datamarts con toda la información necesaria para que puedan ser estudiados por científicos de datos, en busca de nuevos indicadores relevantes a monitorizar o la realización de informes a medida ante situaciones desconocidas.

5. Identificación de tendencias y problemas.

Esta es una de las etapas fundamentales del framwork, ya que nos va a permitir detectar tanto problemas y desafíos como áreas de mejora con el fin de establecer prioridades y tomar medidas correctivas que nos acerquen a la consecución de los objetivos.

a) Identificación de problemas, patrones, anomalías o áreas de mejora. Tras asegurarnos de que la visualización de datos es óptima, es preciso realizar un profundo análisis de estos con el fin de identificar patrones o anomalías que puedan proporcionar insights sobre el rendimiento del sistema y la infraestructura en la nube, que nos permitirán adoptar medidas tendentes a corregir los aspectos deficientes o a potenciar aquellos que permitirán un funcionamiento óptimo del proceso.

b) Priorización de problemas. Una vez detectados los problemas, será necesario realizar una clasificación que nos permita dar prioridad a la resolución de aquellos que

más impactan en la consecución de los objetivos de migración. Además, serán prioritarios siempre por encima del resto los que incidan sobre aspectos como seguridad, funcionalidad o rendimiento.

6. Establecimiento de acciones correctivas.

Una vez que ya tenemos identificados y priorizados tanto los problemas como las áreas de mejora, estamos en disposición de desarrollar planes de acción específicos para resolverlos y contribuir así a la consecución de los objetivos fijados en la fase inicial.

a) Análisis detallado de los problemas. Antes de empezar a desarrollar cualquier tipo de plan de acción, es necesario realizar un minucioso análisis de los problemas detectados, comprender la causa raíz de estos, como afectan al sistema y qué impacto tienen en los objetivos del proyecto. En muchos casos será necesario definir nuevas métricas a monitorizar, contar con el apoyo de los analistas de datos o mantener reuniones con los usuarios del producto para profundizar en las causas raíz.

b) Desarrollo de planes de acción. Una vez comprendida la causa raíz de cada problema, es preciso establecer planes de acción para abordarlos de manera efectiva y rápida. Los planes de acción deben ser muy concretos, especificar los recursos necesarios para su ejecución y en qué plazo deben implementarse.

c) Asignación de plazos y responsabilidades. Cada tarea contemplada en los planes de acción tendrá establecido un plazo de realización realista y que contemple la urgencia del problema a resolver. La priorización de estas tareas deberá siempre atender siempre al impacto y probabilidad de que se repitan las circunstancias que han producido la desviación en la métrica monitorizada. Cada tarea será asignada a un miembro del equipo, quien se responsabilizará de llevarla a cabo en el tiempo y forma establecidos.

e) Seguimiento y monitorización. Una vez establecidos los planes de acción y asignadas las tareas, es necesario realizar un seguimiento para asegurarnos de que los planes se llevan a cabo de forma adecuada. Se establecerán reuniones regulares de seguimiento, actualizaciones de estado y reportes de avance en la implementación de medidas.

d) Ajustes y mejoras. Conforme se van implementando las medidas correctivas que resuelven las incidencias detectadas, es preciso estar abierto a ajustar los planes establecidos en caso necesario. Esos ajustes pueden ser de plazos, responsables, recursos, etc. siempre buscando la mejora en la consecución de objetivos.

7. Evaluación de impacto y retroalimentación.

En esta fase del proceso, se evalúa el impacto que están teniendo las acciones correctivas puestas en marcha en la fase anterior y se recopila y analiza la retroalimentación aportada por usuarios y otras partes interesadas con el fin de evaluar la efectividad de esas medidas.

a) Evaluación de impacto. Se realiza una comparación de métricas actuales y previas, con el fin de evidenciar la mejora que se ha producido en el rendimiento del sistema tras la implementación de las acciones correctivas.

b) Recopilación y análisis de retroalimentación. Se recopila toda la información posible de usuarios, equipo de desarrollo, dirección de proyecto, etc. para posteriormente realizar un minucioso análisis que nos permita identificar nuevos patrones, tendencias o áreas de mejora. Es básico que se analice tanto la información negativa como positiva para así tener el espectro completo de la retroalimentación.

8. Iteración y mejora continua.

Esta última fase de la metodología se centra en repetir el proceso de monitorización, análisis e implementación de acciones correctivas para mantener y regular el rendimiento del sistema.

a) Programación de revisiones periódicas. Se debe establecer un cronograma regular de revisiones del rendimiento del sistema buscando un intervalo de tiempo que se adecúe a las características del proyecto.

b) Recopilación de datos continua. Durante cada ciclo de iteración se continúa recopilando datos, para eso es necesario comprobar periódicamente que las herramientas configuradas para este fin tienen un funcionamiento óptimo.

c) Repetición de proceso desde fase 4. El proceso se repetirá en cada iteración desde la fase de análisis y visualización de datos y pasará por cada una de las fases comentadas con el fin de mantener y mejorar constantemente el rendimiento del sistema.

4. MÉTRICAS METODOLOGÍA SWIM

En términos de métricas de rendimiento, se han definido diversas medidas para evaluar la eficiencia y la velocidad de los sistemas de software. Esto incluye métricas como el tiempo de respuesta, que mide el tiempo transcurrido entre una solicitud y su respuesta correspondiente, y la velocidad de procesamiento, que cuantifica la cantidad de tareas que un sistema puede completar en un período de tiempo determinado.

Además, la utilización de recursos, que evalúa cómo se utilizan los recursos del sistema, como la CPU y la memoria, también se considera fundamental en la medición del rendimiento del software.

En cuanto a las métricas de fiabilidad, se han propuesto medidas para evaluar la estabilidad y la confiabilidad de los sistemas de software. Estas métricas incluyen la tasa de errores, que cuenta el número de errores o fallos encontrados durante un período de tiempo específico, y la disponibilidad del sistema, que mide el tiempo durante el cual el sistema está disponible y operativo. Además, la capacidad de recuperación del sistema, que evalúa la capacidad de un sistema para recuperarse de fallos o interrupciones, también es crucial en la evaluación de la fiabilidad del software.

En lo que respecta a las métricas de escalabilidad, se han propuesto medidas para evaluar la capacidad de los sistemas de software para adaptarse a cargas de trabajo variables y crecientes. Estas métricas incluyen la capacidad de procesamiento, que cuantifica la cantidad de trabajo que un sistema puede manejar en un momento dado, y la escalabilidad lineal, que mide cómo varía el rendimiento del sistema al aumentar la carga de trabajo. Además, la capacidad de distribución, que evalúa la capacidad de un sistema distribuido para escalar horizontalmente agregando más nodos, también se considera importante en la medición de la escalabilidad del software.

Finalmente, en cuanto a las métricas de usabilidad, se han propuesto medidas para evaluar la facilidad de uso y la satisfacción del usuario de los sistemas de software. Estas métricas incluyen la facilidad de aprendizaje, que evalúa la facilidad con la que los usuarios pueden aprender a utilizar el sistema, y la satisfacción del usuario, que mide la percepción general de los usuarios sobre la usabilidad y la utilidad del sistema. Además, la accesibilidad del sistema, que evalúa la facilidad con la que los usuarios con discapacidades pueden acceder y utilizar el sistema, también se considera fundamental en la evaluación de la usabilidad del software.

Antes de entrar en el detalle de las métricas, hay que tener en cuenta que, si bien se han propuesto numerosas métricas para medir diferentes aspectos del desempeño del software, es importante reconocer la necesidad de un enfoque integral que abarque múltiples dimensiones. Al considerar una variedad de métricas que evalúen el rendimiento, la fiabilidad, la escalabilidad y la usabilidad del software, los equipos de

producto pueden obtener una imagen completa y detallada de su funcionamiento, lo que les permite identificar áreas de mejora y tomar medidas correctivas para optimizar su desempeño y calidad.

Algunas métricas propuestas:

Las métricas utilizadas para evaluar el rendimiento de las aplicaciones pueden variar dependiendo de si se trata de aplicaciones legacy o aplicaciones basadas en la nube.

Aquí hay algunas diferencias clave en la aplicación de métricas en cada caso:
Aplicaciones Legacy:

1. Infraestructura estática: Las aplicaciones legacy a menudo se ejecutan en infraestructuras de hardware dedicadas y estáticas. Esto significa que la escalabilidad y la capacidad de respuesta pueden ser limitadas y predecibles.

2. Optimización de recursos: Las métricas de rendimiento en aplicaciones legacy a menudo se centran en la optimización de recursos de hardware, como el uso de CPU, memoria y almacenamiento. Se prestan más atención a métricas como la utilización de CPU, el tiempo de respuesta del servidor y el uso de memoria.

3. Mantenimiento del rendimiento a largo plazo: En las aplicaciones legacy, las métricas de rendimiento a menudo se utilizan para monitorizar el rendimiento a largo plazo y garantizar que el sistema siga siendo eficiente y confiable a medida que evoluciona con el tiempo.

4. Enfoque en la estabilidad y confiabilidad: Dado que las aplicaciones legacy a menudo tienen una arquitectura monolítica y pueden ser difíciles de modificar, las métricas de rendimiento se centran en garantizar la estabilidad y confiabilidad del sistema en lugar de la escalabilidad y la flexibilidad.

Aplicaciones en la Nube:

1. Escalabilidad dinámica: Las aplicaciones en la nube aprovechan recursos informáticos escalables y elásticos, lo que permite una escalabilidad dinámica según la demanda. Las métricas de rendimiento se centran en la capacidad de la aplicación para escalar horizontal y verticalmente para manejar cambios en la carga de trabajo.

2. Disponibilidad y tolerancia a fallos: Dado que las aplicaciones en la nube pueden estar sujetas a interrupciones y fallos en la infraestructura subyacente, las métricas de rendimiento se centran en la disponibilidad, la resistencia y la tolerancia a fallos del sistema.

3. Optimización de costes: Las métricas en las aplicaciones en la nube también pueden incluir el coste de la infraestructura y el rendimiento en relación con los costes. Esto implica monitorizar métricas de rendimiento como el coste por transacción o el coste por usuario activo para garantizar una optimización efectiva de los recursos.

4. Adaptabilidad y flexibilidad: Las aplicaciones en la nube se benefician de una arquitectura más modular y orientada a servicios, lo que permite una mayor adaptabilidad y flexibilidad. Las métricas de rendimiento pueden centrarse en la agilidad del desarrollo, la frecuencia de implementación y la capacidad de respuesta a los cambios en los requisitos del negocio.

En conclusión, mientras que las aplicaciones legacy se centran en la estabilidad y eficiencia de los recursos, las aplicaciones en la nube están más orientadas hacia la escalabilidad, la disponibilidad y la adaptabilidad. Las métricas de rendimiento se adaptan en consecuencia para abordar las necesidades específicas de cada tipo de aplicación. Si nuestro producto es la evolución de una solución software legacy, y estamos cambiando el paradigma de interacción con el usuario, será recomendable poder disponer de cómo se comportaban los indicadores en el sistema legacy, en comparación con cómo se comportan los mismos en el sistema cloud, con el objetivo de poder evidenciar los puntos de mejora y minimizar el impacto en los usuarios finales durante todo el proceso de transición hacia una nueva forma de relación.

Nuestra propuesta de métricas se basa en la comprensión de que el desempeño del software es multidimensional y requiere una evaluación integral para garantizar su calidad y eficacia.

Por lo tanto, hemos identificado cuatro dimensiones principales del desempeño del software: rendimiento, fiabilidad, escalabilidad y usabilidad. A continuación, presentamos las métricas específicas propuestas para cada una de estas dimensiones:

MÉTRICAS DE RENDIMIENTO

Al medir y monitorizar estas métricas de rendimiento de manera regular, los equipos pueden identificar cuellos de botella, detectar problemas de rendimiento y tomar medidas para optimizar el desempeño del sistema software.

Es importante tener en cuenta que los objetivos variarán según las características específicas de la aplicación, las expectativas del usuario y las limitaciones del entorno de la nube. Además, es fundamental monitorizar regularmente el rendimiento de la aplicación y ajustar los objetivos según sea necesario para garantizar un funcionamiento óptimo a lo largo del tiempo.

Resumiendo, controlar estas métricas de rendimiento u otras complementarias es fundamental para garantizar un funcionamiento eficiente, confiable y receptivo del sistema de software.

1.Tiempo de respuesta (Response Time):

El tiempo de respuesta hace referencia al tiempo (en milisegundos) que transcurre desde que se emite una solicitud hasta que se recibe la respuesta correspondiente del sistema. Un tiempo de respuesta bajo ayuda a mantener la atención del usuario y proporciona una sensación de fluidez en la interacción con la aplicación, lo que mejora la experiencia del usuario y es crucial para garantizar su satisfacción y la percepción de un sistema ágil y receptivo. Las aplicaciones con tiempos de respuesta largos pueden resultar frustrantes y desalentar a los usuarios.

Tiempo de respuesta = Tiempo de finalización de la solicitud - Tiempo de inicio de la solicitud.
Objetivo: Menos de 500 milisegundos para solicitudes críticas del usuario.

2. Tasa de procesamiento (Throughput Rate):

La tasa de procesamiento, medida en transacciones completadas por unidad de tiempo, es esencial para evaluar la capacidad de un sistema para gestionar un volumen dado de solicitudes o transacciones. Una alta tasa de procesamiento asegura que el sistema pueda satisfacer la demanda de los usuarios sin retrasos significativos. Se calcula dividiendo el número total de transacciones completadas por el tiempo total, y se expresa típicamente en unidades como transacciones por segundo (tps) o solicitudes por minuto (rpm). Una tasa de procesamiento alta indica una capacidad robusta del sistema para manejar cargas de trabajo. Por ejemplo, el objetivo puede ser alcanzar una capacidad de al menos 1000 transacciones por segundo, lo que garantiza que la aplicación pueda mantenerse efectiva incluso durante periodos de alta demanda de usuarios.

Tasa de procesamiento = Número total de transacciones completadas / Tiempo total
Objetivo: Capacidad para manejar al menos 1000 transacciones por segundo.

3. Utilización de recursos (Resource Utilization)

La utilización de recursos se refiere a la proporción de recursos del sistema utilizados (%) en un momento dado, como la CPU, la memoria y el ancho de banda de red. Es esencial para maximizar el rendimiento y la escalabilidad del sistema. Monitorizar esta utilización ayuda a identificar posibles cuellos de botella y optimizar la asignación de recursos. Se calcula como el porcentaje de recursos utilizados respecto al total disponible. Mantener la utilización por debajo del 70% de su capacidad máxima es crucial para garantizar un rendimiento óptimo y evitar sobrecargas, asegurando la escalabilidad del sistema frente a aumentos repentinos en la carga de trabajo.

Utilización de recursos = (Recursos utilizados / Recursos totales) * 100%.
Objetivo: Mantener la utilización de recursos por debajo del 70% de su capacidad máxima.

4. Latencia (Latency)

La latencia se refiere al tiempo que tarda una solicitud (ms) en viajar desde su origen hasta su destino, incluyendo el tiempo de procesamiento. Es crucial para la comunicación eficiente entre los componentes de un sistema distribuido. Una baja latencia es fundamental para garantizar una comunicación rápida, especialmente en aplicaciones en tiempo real o de alta velocidad. Se calcula restando el tiempo de inicio de la solicitud del tiempo de finalización. Se mide típicamente en milisegundos (ms) o segundos (s). Mantener la latencia por debajo de 100 milisegundos es crucial para una comunicación fluida en la aplicación en la nube, evitando retrasos perceptibles que puedan afectar la experiencia del usuario.

Latencia = Tiempo de finalización de la solicitud - Tiempo de inicio de la solicitud.
Objetivo: Mantener la latencia de red por debajo de 100 milisegundos.

5. Capacidad de concurrencia (Concurrency Capacity)

La capacidad de concurrencia se refiere a la cantidad máxima de solicitudes o transacciones que un sistema puede manejar simultáneamente sin una degradación significativa del rendimiento. Es esencial para garantizar que el sistema pueda manejar picos de carga sin experimentar tiempos de espera excesivos o fallos del sistema debido a la sobrecarga. Se mide en el número máximo de conexiones simultáneas admitidas. Una alta capacidad de concurrencia es crucial para mantener un rendimiento consistente bajo cargas de trabajo pesadas. El objetivo es poder manejar al menos 1000 conexiones simultáneas, lo que garantiza que la aplicación pueda manejar múltiples solicitudes concurrentes sin degradación del rendimiento ni tiempos de espera excesivos.

Capacidad de concurrencia = Número máximo de conexiones simultáneas admitidas.
Objetivo: Capacidad para manejar al menos 1000 conexiones simultáneas

MÉTRICAS DE FIABILIDAD

Al establecer objetivos y monitorizar estas métricas de fiabilidad, los equipos pueden garantizar la estabilidad, la disponibilidad y la confiabilidad del sistema, lo que contribuye a proporcionar una experiencia de usuario satisfactoria y mantener la confianza del cliente.

1. Tasa de errores (Error Rate)

La tasa de errores es el porcentaje de transacciones o eventos que resultan en un error o fallo en el sistema durante un período de tiempo específico. Es esencial para evaluar la fiabilidad del sistema y garantizar una experiencia de usuario satisfactoria. Se calcula dividiendo el número total de errores entre el número total de transacciones y se expresa como un porcentaje. Mantener una tasa de errores por debajo del 0.1% es crucial para asegurar una alta fiabilidad del sistema, lo que contribuye a la confianza del usuario y la satisfacción del cliente.

Tasa de errores = (Número total de errores / Número total de transacciones)* 100%
Objetivo: Mantener la tasa de errores por debajo del 0.1%.

2. Disponibilidad del sistema (System Availability)

La disponibilidad del sistema es el porcentaje de tiempo en que el sistema está disponible y operativo para los usuarios. Es crítica para garantizar la continuidad del servicio y la satisfacción del usuario. Se calcula dividiendo el tiempo de funcionamiento entre el tiempo total y se expresa como un porcentaje. Lograr una disponibilidad del sistema del 99.9% (menos de 8.76 horas de tiempo de inactividad por año) es fundamental para asegurar una mayor confiabilidad del sistema y evitar interrupciones en el negocio.

Disponibilidad del sistema = (Tiempo de funcionamiento / Tiempo total) * 100%.
Objetivo: Lograr una disponibilidad del sistema del 99.9% (menos de 8.76 horas de tiempo de inactividad por año).

3. Tiempo medio entre fallos (Mean Time Between Failures, MTBF)

El MTBF (Tiempo Medio entre Fallas) es el tiempo promedio que transcurre entre fallos del sistema, una medida clave de su fiabilidad. Un MTBF alto indica una mayor resistencia ante fallos. Se calcula dividiendo el tiempo total de operación entre el número total de fallos, y se mide en horas. El objetivo es alcanzar un MTBF de al menos 10,000 horas, lo que reduce la probabilidad de tiempos de inactividad no planificados y mejora la experiencia del usuario al garantizar un funcionamiento continuo y sin interrupciones.

MTBF = Tiempo total de operación / Número total de fallos.
Objetivo: Lograr un MTBF de al menos 10,000 horas.

4.Tiempo medio de recuperación (Mean Time to Recovery, MTTR)

El MTTR (Tiempo Medio para Reparar) es el tiempo promedio que tarda el sistema en recuperarse después de un fallo o interrupción. Es una medida crucial de la capacidad del sistema para recuperarse de fallos. Un MTTR corto es esencial para minimizar el tiempo de inactividad y mantener la continuidad del servicio. Se calcula dividiendo el tiempo total de inactividad entre el número total de recuperaciones, y se mide típicamente en minutos o segundos. El objetivo es mantener un MTTR inferior a 30 minutos, lo que garantiza una rápida restauración del servicio y minimiza el impacto de las interrupciones en la operatividad del negocio.

MTTR = Tiempo total de inactividad / Número total de recuperaciones.
Objetivo: Mantener un MTTR inferior a 30 minutos.

5. Redundancia y tolerancia a fallos.

La redundancia y la tolerancia a fallos son estrategias de diseño que implican duplicar componentes críticos o implementar mecanismos que permitan que un sistema continúe funcionando incluso si ocurren fallos en algunos de sus componentes. Son cruciales para garantizar la disponibilidad y la fiabilidad del sistema en entornos críticos, minimizando los tiempos de inactividad y protegiendo los datos importantes. No hay una fórmula específica para estas estrategias, ya que dependen de las necesidades del sistema. La redundancia duplica componentes críticos, mientras que la tolerancia a fallos permite al sistema detectar, mitigar y recuperarse de fallos automáticamente. El objetivo es garantizar la disponibilidad continua del sistema y minimizar el impacto de los fallos en el negocio, mediante la implementación de redundancia y mecanismos de tolerancia a fallos. Estas estrategias son esenciales en entornos donde la disponibilidad y la fiabilidad son críticas, como sistemas financieros o médicos, reduciendo los tiempos de inactividad y manteniendo la confianza del cliente.

MÉTRICAS DE ESCALABILIDAD

ALas métricas de escalabilidad tienen un papel básico en cualquier migración al cloud, pues nos van a permitir evaluar la capacidad de los sistemas para adaptarse a las necesidades del sistema bajo situaciones de estrés.

1. Capacidad de autoescalado

La capacidad de autoescalado es la habilidad de una aplicación o infraestructura en la nube para ajustar automáticamente sus recursos en respuesta a cambios en la carga de trabajo, garantizando así un rendimiento óptimo. Permite a las aplicaciones en la nube manejar picos de tráfico sin intervención manual, mejorando la experiencia del usuario y optimizando los costes operativos al ajustar los recursos según la demanda. No tiene una fórmula específica ni unidades de medida. Representa la capacidad de adaptación de la aplicación en la nube, siendo crucial para mantener un rendimiento constante ante variaciones en la demanda. El objetivo es garantizar que la aplicación pueda escalar automáticamente en respuesta a cambios en la carga de trabajo para mantener un rendimiento estable y una experiencia de usuario óptima, minimizando los costes operativos y de infraestructura. La capacidad de autoescalado mejora la eficiencia operativa al evitar la subutilización o la saturación de recursos, y garantiza que la aplicación pueda manejar tanto cargas de trabajo inesperadas como fluctuaciones predecibles.

2. Tiempo de respuesta bajo carga creciente:

El tiempo de respuesta bajo carga creciente es el tiempo que tarda la aplicación en responder a las solicitudes de los usuarios a medida que aumenta la carga de trabajo. Evaluar este tiempo asegura que la aplicación pueda manejar un aumento en la demanda sin experimentar una degradación significativa del rendimiento. No tiene una fórmula específica, pero se mide en milisegundos (ms) o segundos (s). Indica la capacidad de la aplicación para mantener un tiempo de respuesta aceptable incluso bajo una mayor carga de trabajo. El objetivo es mantener un tiempo de respuesta constante independientemente del aumento de la carga de trabajo. Un tiempo de respuesta bajo carga creciente garantiza una experiencia de usuario consistente y satisfactoria, lo que ayuda a retener a los usuarios y mejora la reputación de la aplicación.

3. Elasticidad

La elasticidad se refiere a la capacidad de una aplicación en la nube para aumentar o disminuir dinámicamente sus recursos en función de la demanda, asegurando que siempre haya suficientes recursos disponibles para mantener un rendimiento óptimo. Garantiza que la aplicación pueda escalar rápidamente para manejar picos de tráfico inesperados o fluctuaciones en la demanda, optimizando así los costes operativos y mejorando la experiencia del usuario. No tiene una fórmula específica ni unidades de medida, pero representa la capacidad de adaptación y respuesta de la aplicación en la nube ante cambios en la carga de trabajo. El objetivo es garantizar que la aplicación pueda aumentar o disminuir sus recursos según sea necesario para mantener un rendimiento estable y eficiente. La elasticidad proporciona agilidad y flexibilidad a la aplicación, permitiéndole adaptarse rápidamente a las condiciones cambiantes del mercado y satisfacer las necesidades del negocio de manera eficiente.

4. Disponibilidad y redundancia

La disponibilidad y redundancia se refieren a la capacidad de una aplicación en la nube para mantener la disponibilidad del servicio incluso en caso de fallos de componentes individuales, mediante la duplicación de recursos críticos. Garantizan que la aplicación esté siempre disponible para los usuarios, incluso durante fallos de infraestructura, actualizaciones o mantenimiento. No hay una fórmula específica para esto, pero se mide en porcentaje (%), representando la confiabilidad y estabilidad del servicio proporcionado por la aplicación en la nube. El objetivo es mantener un alto nivel de disponibilidad del servicio, minimizando los tiempos de inactividad y garantizando la continuidad del negocio. La disponibilidad y redundancia aseguran que la aplicación pueda mantenerse operativa en todo momento, evitando interrupciones en el servicio que podrían afectar negativamente la experiencia del usuario y la reputación de la empresa.

5. Coste de escalabilidad

El coste de escalabilidad es el coste asociado con el escalado de recursos en la nube para satisfacer la demanda de la aplicación. Evaluar este coste es fundamental para garantizar que la aplicación pueda escalar de manera rentable y eficiente, optimizando el uso de recursos y minimizando los costes operativos. Se mide en la moneda local, como dólares o euros, y representa el impacto económico del escalado de recursos en la nube para la aplicación. El objetivo es minimizar los costes asociados con el escalado de recursos en la nube, garantizando que la aplicación pueda crecer de manera rentable. Evaluar el coste de escalabilidad ayuda a optimizar el uso de recursos y tomar decisiones informadas sobre la arquitectura y el dimensionamiento de la aplicación, asegurando una utilización eficiente de los recursos y maximizando el retorno de la inversión.

MÉTRICAS DE USABILIDAD

Estas métricas de usabilidad proporcionan una visión detallada de la experiencia del usuario y la efectividad del sistema en términos de eficiencia, efectividad, satisfacción y facilidad de uso. Al medir y mejorar estos aspectos, los diseñadores y desarrolladores pueden crear sistemas más efectivos y satisfactorios para los usuarios.

Al considerar estas métricas propuestas en conjunto, los equipos pueden obtener una evaluación completa y detallada del desempeño del software, lo que les permite identificar áreas de mejora y tomar medidas correctivas para optimizar su calidad y eficacia.

1. Tiempo de tarea (Task Time)

El tiempo de tarea es el tiempo necesario para que un usuario complete una tarea específica dentro del sistema, desde el inicio hasta la finalización. Evaluar el tiempo de tarea ayuda a comprender la eficiencia del sistema y la experiencia del usuario al realizar acciones concretas. No se aplica una fórmula específica, simplemente se registra el tiempo tomado para completar la tarea, y se mide en segundos (s) o minutos (min).

Cuanto menor sea el tiempo de tarea, más eficiente y satisfactoria será la experiencia del usuario al interactuar con el sistema. El objetivo es reducir el tiempo de tarea para mejorar la eficiencia y la productividad del usuario. Al analizar y optimizar el tiempo de tarea, se puede mejorar la experiencia del usuario, lo que a su vez puede conducir a una mayor satisfacción y retención de usuarios.

2. Tasa de éxito (Success Rate)

La tasa de éxito es el porcentaje de usuarios que completan una tarea específica con éxito dentro del sistema. Esta métrica proporciona una medida directa de la efectividad del sistema y su capacidad para ayudar a los usuarios a lograr sus objetivos. Se expresa en porcentaje (%) y una alta tasa de éxito indica que el sistema es fácil de usar y que los usuarios pueden completar sus tareas sin dificultades significativas. El objetivo es maximizar la tasa de éxito para garantizar que la mayoría de los usuarios puedan completar sus tareas de manera satisfactoria. Una alta tasa de éxito es fundamental para asegurar la utilidad y la eficacia del sistema, lo que contribuye a la satisfacción y la fidelidad del usuario.

Tasa de éxito = (Número de usuarios que completaron la tarea con éxito / Número total de usuarios que intentaron la tarea) * 100%.

3. Tasa de error (Error Rate):

La tasa de error es el porcentaje de veces que los usuarios cometen errores al realizar una tarea en el sistema. Su evaluación permite identificar áreas de confusión o dificultad dentro del sistema que pueden requerir mejoras en el diseño o la usabilidad.
Una baja tasa de error indica que el sistema es claro y fácil de entender, mientras que, si esta tasa es elevada, puede indicar que hay problemas de usabilidad. Reducir

la tasa de error puede aumentar la confianza del usuario en el sistema y disminuir la frustración, lo que conduce a una experiencia más positiva y satisfactoria. Obviamente el objetivo será mantener esta tasa en los niveles más bajos posibles.

Tasa de error = (Número de errores cometidos por los usuarios / Número total de intentos de tarea) * 100%.

4. Facilidad de aprendizaje (Ease of Learning):

La facilidad de aprendizaje se refiere al tiempo y esfuerzo que requiere un usuario para aprender a utilizar el sistema por primera vez. Si obtenemos una fácil curva de aprendizaje, sabemos que el usuario es capaz de utilizar el sistema de forma rápida, reduciendo así la resistencia al cambio y mejora la adopción del usuario.

Se medirá registrando el tiempo que un usuario necesita para poder utilizar el sistema de manera eficiente. Cuanto menos tiempo y esfuerzo se requiera para aprender a usar el sistema, más fácil será incorporarlo a los flujos de trabajo.

5. Satisfacción del usuario (User Satisfaction):

La satisfacción del usuario es la medida en que los usuarios se muestran conformes con la experiencia general de usar el sistema. Este indicador, nos da una medida sobre la calidad y utilidad percibida del sistema, lo que puede tener un impacto significativo en el éxito a largo plazo del producto o servicio. A la hora de cuantificarlo, lo podemos medir con una escala de puntuación (de 1 a 5, por ejemplo).
Una alta satisfacción del usuario indica que el sistema cumple con sus expectativas y necesidades, mientras que una baja satisfacción puede indicar problemas que requieren atención.

6. Densidad de información (Information Density):

La densidad de información se refiere a la cantidad de información presentada en una página o pantalla del sistema. Una densidad de información equilibrada facilita la comprensión y navegación del sistema, evitando la sobrecarga cognitiva y mejorando la eficiencia del usuario.

Una alta densidad de información puede abrumar al usuario, mientras que una baja densidad puede resultar en una experiencia vacía o incompleta. Por tanto, se tratará de mantener una densidad de información equilibrada que ayude a mejorar la legibilidad y usabilidad del sistema, facilitando a los usuarios encontrar y procesar la información relevante de una manera eficiente.

5. CONCLUSIONES

En este artículo, hemos abordado inicialmente el desarrollo del framework integral SWIM, especialmente diseñado para medir diferentes aspectos del desempeño del software con el fin de obtener una visión completa de su calidad y eficacia.

Dicha metodología se centra en establecer objetivos específicos y definir métricas, implementación de herramientas de monitorización, recopilación y almacenamiento de datos para posteriormente analizarlos y detectar tendencias e incidencias que derivan en una serie de medidas correctivas. Finalmente, se evalúa el impacto de esas medidas y se vuelve a iniciar el proceso con el fin de que la mejora sea continua.

A continuación, se han presentado un conjunto integral de métricas para medir el desempeño del software en la nube, las cuales pueden aplicarse a lo largo del ciclo de vida del software, desde el diseño y desarrollo hasta la implementación y mantenimiento. Al abarcar múltiples dimensiones, estas métricas proporcionan una evaluación completa y detallada del software, lo que permite siguiendo la metodología SWIM, que los equipos identifiquen errores y áreas de mejora para garantizar así la calidad y eficacia del producto final.

6.REFERENCIAS

Abbas, Z. (2023). Cloud Migration Strategies: Moving Applications and Workloads to the Cloud. Department of Computer Science, University of South Coria.
Al-Qutaish, R. (2006). Performance Metrics for Software Development. Information and Software Technology, 48(4), 309-319.

Caglayan, B., Baykan, Ö. K., & Kocak, M. (2013). Comprehensive Evaluation of Software Metrics in Predicting Maintainability. Journal of Systems and Software, 86(6), 1538-1556.

Chauan, M. A. y Babar, M.A. (2012) Towards Process Support for Migrating Applications to Cloud Computing. International Conference on Cloud and Service Computing. EEE Computer Society Press. Washington D.C.

Fahmideh, M. et alter (2017) Challenges in migrating legacy software systems to the cloud—an empirical study. Information System Journal. Elsevier V.B. London.

Gray, J., & MacDonell, S. G. (1997). Predicting Software Quality Using Software Function Point Metrics. IEEE Transactions on Software Engineering, 23(8), 529-540.

Imeri, F. y Memeti, A. (2018) Review of cloud migration strategies and providers – an IaaS migration strategy. UBT International Conference 107. University of Business and Technology in Kosovo.

Lakhotia, K., Aggarwal, K. K., & Kaur, A. (2015). Comprehensive Review on Software Metrics. Journal of Information Systems and Software Engineering, 1(1), 1-8.

Mendes, E., Mosley, N., Counsell, S., & Hierons, R. M. (2002). Predicting Software Quality Through Software Complexity Metrics. Information and Software Technology, 44(15), 871-882

Melo, W. L., Ferreira, R. M., & Frota, Y. L. (2007). A Comprehensive Framework for Software Product Metrics. Information and Software Technology, 49(10), 1080-1099.

Muhamad, H. et alter (2023) Legacy systems to cloud migration: A review from the architectural perspective. Journal of Systems and Software. Vol. 202. Elsevier V.B. London.

Pulkkinen, M. (2014) Cloud migration strategy factors and migration processes. Faculty of Science. University of Helsinki. Helsinki.

Tantry, H.S. et alter (2020). Impact Analysis of Legacy System Migration to the Cloud Environment: A Focused Study. International Journal of Advanced Trends in Computer Science and Engineering. Vol. 9. N°1. Tamilnadu. 134-141
Zalazar, A.S. et alter (2014). Migración de Sistemas Heredados a Cloud Computing. INGAR (UTN-CONICET) 15th Argentine Symposium on Software Engineering, ASSE 2014. Santa Fe. 66-79.

clave i

Software solutions for business