Diseño de pruebas de carga

Breve descripción:

En este componente formativo, se abordará el diseño de pruebas de carga con el objetivo de validar características específicas que representan el rendimiento y las prestaciones ofrecidas al usuario final de una aplicación o plataforma. Esta se somete a prueba con un gran volumen de peticiones o solicitudes, las cuales simulan una carga de trabajo representativa de un escenario previsto durante la operación diaria en un ambiente productivo.

Abril 2024

Tabla de contenido

Introducción

Bienvenidos a este componente formativo, en el que se abordará el diseño de pruebas de carga. En la industria del "software", existe una alta demanda de personal capacitado para determinar las características de rendimiento que presentan los productos o las que se desearían que tuvieran.

Este personal calificado es capaz de identificar y evaluar atributos de calidad del "software" relacionados con la operación de los servicios de teleinformática, una vez que están disponibles como servicios tecnológicos. Esto permite determinar, antes de que los sistemas sean implementados en producción, su capacidad, así como su habilidad para atender a la demanda de un número específico de usuarios.

De este modo, el trabajo en pruebas de rendimiento, un subconjunto de la ingeniería de pruebas, representa una oportunidad de negocio que pocos conocen y explotan. Esta área del conocimiento en el desarrollo de aplicaciones puede ofrecer estabilidad a largo plazo y una carrera sólida a quienes decidan explorarla.

Pero, ¿qué son las pruebas de rendimiento? ¿Cuáles son los procesos que buscan verificar la eficiencia, eficacia, fiabilidad, escalabilidad, velocidad y calidad de respuesta de los sistemas informáticos bajo condiciones de trabajo específicas?

Para responder a estas preguntas, le invitamos a revisar este componente formativo, en el que identificará los conceptos básicos sobre evaluación de escalabilidad, instalación y configuración de herramientas de pruebas de rendimiento, y la metodología para realizar esta evaluación. De esta manera, se generan indicadores

de calidad para la toma de decisiones de los equipos de desarrollo o la implementación de los sistemas tecnológicos.

1. Evaluación comparativa de escalabilidad

Una de las etapas del proceso de desarrollo que genera opiniones divididas en la construcción de una aplicación es la relacionada con las pruebas del sistema. Este paso es crucial para asegurar la calidad del "software"; sin embargo, la mayoría de estas pruebas se centran en validar y certificar componentes, módulos o incluso flujos del proceso en un entorno controlado, disponiendo de equipos con gran capacidad a nivel de "hardware" que soportan la solución de "software".

Es importante tener en cuenta que este tipo de "testing" minimiza el aseguramiento de la calidad del servicio, ya que no simula el posible potencial de carga y la aleatoriedad de las peticiones que puede recibir la aplicación en un entorno productivo, lo que imposibilita encontrar cuellos de botella en algún componente, módulo o flujo del aplicativo, haciendo que las oportunidades de mejora y escalabilidad disminuyan y los problemas asociados al rendimiento aumenten el tiempo, dinero y la improvisación de soluciones para el equipo de desarrollo. En este punto se debe realizar un adecuado plan de pruebas de carga para evaluar la escalabilidad y así poder brindar soluciones adecuadas en el momento correcto.

Antes de profundizar en los aspectos centrales de este tema, es importante comprender algunos conceptos fundamentales relacionados con los requisitos técnicos del diseño de cargas. Revisemos, entonces, los términos de escalabilidad y evaluación comparativa.

¿Qué es la escalabilidad?

Según Meier et al. (2007), la escalabilidad es la capacidad de una aplicación para manejar carga de trabajo adicional, sin afectar negativamente el rendimiento, añadiendo recursos tales como el procesador, la memoria y la capacidad de almacenamiento.

Este concepto, a menudo percibido como abstracto, está intrínsecamente vinculado al diseño de la arquitectura de "software", ya que un sistema bien diseñado no enfrentará problemas de escalabilidad. Por lo tanto, es posible concluir que la escalabilidad no es un atributo que se pueda configurar en el sistema posteriormente, sino que depende de la implementación y del diseño inicial del mismo.

La escalabilidad representa un punto crítico en el crecimiento de un proyecto, ya sea para aumentar el número de usuarios atendidos simultáneamente o para ampliar las funcionalidades ofrecidas. Esto implica que un líder debe evaluar dos posibles enfoques, representados por los siguientes ejemplos:

- **Mejorar en combinación el "hardware" y "software"**. Ejemplo: pasar de almacenar en un servidor de base de datos local con MySQL versión 5.0 a un almacenamiento en Amazon Web Services con Aurora DB.
- **Aumentar las capacidades y la potencia del "hardware"**. Ejemplo: pasar de una máquina con 16 GB de memoria RAM y un disco duro mecánico de 500 GB a una máquina de 32 GB de RAM y un disco duro de estado sólido de 1 TB.

Es importante conocer que la escalabilidad también se clasifica en vertical y horizontal, tal como se explica a continuación:

Escalabilidad vertical

Se trata básicamente de añadir recursos en un solo componente o nodo "hardware" dentro del sistema, aumentando más memoria RAM, más capacidad de almacenamiento del disco duro o agregando en nuevos "slot" componentes de los anteriormente mencionados.

- **Ventajas**: el "software" no se ve afectado por el proceso, además la arquitectura inicial no necesita realizar cambio alguno y el impacto en codificación es mínimo o nulo.
- **Desventajas**: los componentes "hardware" tienen una limitación de fábrica, es decir, llegará el momento en que no podrán mejorar, aún más, la memoria RAM.
- **A nivel económico**: el costo en "hardware" de alta prestación se aumenta, teniendo en cuenta que cuando un dispositivo llega al umbral máximo se tratará como tecnología de punta, lo que en ocasiones no es asumible por un proyecto.

Escalabilidad horizontal

Este tipo de escalabilidad significa agregar más nodos o componentes con las mismas características que el sistema ya tiene, por lo general, este nuevo integrante funciona a modo de respaldo y/o espejo.

- **Ventajas**: es posible crecer constantemente y con un bajo costo económico, además de que el mantenimiento es simplificado por lo simple de los nodos o componentes del "cluster".
- **Desventajas**: este modelo implica un cambio sustancial en el diseño y la arquitectura planteada para un único componente o nodo, obligando a

modificar código y, en ocasiones, a reestructurar completamente la solución "software", lo que presume un gran esfuerzo y tiempo por parte del equipo.

Para continuar, se indaga sobre el concepto Evaluación comparativa:

"De acuerdo con Mejer et al. (2007), la evaluación comparativa es el proceso de comparar el rendimiento de un sistema con una línea de base que se ha establecido internamente o con un estándar de la industria avalado por alguna otra organización." **Mejer et al. (2007)**

Se pueden emplear dos formas de establecer la evaluación comparativa de escalabilidad: el Acuerdo de Nivel de Servicio (SLA) y Comparativa con una línea base. Veamos en qué consiste cada una:

Acuerdo de Nivel de Servicio SLA

Los Acuerdos de Nivel de Servicio ("Service Level Agreements", SLA) representan una de las primeras maneras de definir las metas que los sistemas deben alcanzar. Pero, ¿qué es un SLA?, ¿qué determina? y ¿cuáles son sus ventajas? Para responder a estas preguntas, lo invitamos a revisar el siguiente recurso:

¿Qué es SLA?

Es un contrato celebrado entre un proveedor de un servicio y su cliente, que tiene como objetivo establecer niveles (valores) para la calidad con la cual se prestará el servicio ofrecido en contraparte. Dentro del área del "software", es una herramienta que brinda a las partes unos términos aprobados a nivel de calidad en la experiencia del usuario al usar un producto o servicio, en aspectos como tiempo de respuesta, ayudas

en la plataforma (documentación fácil y disponible), disponibilidad horaria 7 por 24, mesa de ayuda (“HelpCenter”, mesa de Ayuda).

¿Qué establece?

Un objetivo es aquella meta a la cual se desea llegar, como por ejemplo, incrementar la cantidad de visitas o aumentar el tiempo de permanencia de un usuario en la navegación de un portal. Por otro lado, el “target” se expresa de forma numérica, es el estado al cual se desea llegar y está directamente relacionado con los objetivos propuestos, por ejemplo, se desea llegar a 50.000 visitas diarias.

¿Qué beneficios ofrece?

Según Mazano (2017) un SLA puede definir un punto del cual partir para la mejora continua en los procesos y procedimientos, ya que ofrece una adecuada medición en términos comunes entre ambas partes plasmadas en el documento a fin de realizar una renegociación y mejorar cada punto ya estipulado brindando una mejor experiencia de usuario y calidad en el servicio.

Parámetros habituales

Un acuerdo de nivel de servicio puede incluir una amplia variedad de parámetros que se alinean con un objetivo específico del nivel de servicio. A continuación, se ofrecen algunos ejemplos:

- Tasa de abandono. Porcentaje de solicitudes abandonadas mientras esperaban recibir atención.
- Tiempo medio de atención. Tiempo medio normalmente medido en segundos, utilizado para que la mesa de ayuda responda una solicitud.

- Factor del tiempo de nivel de servicio. Porcentaje de solicitudes respondidas en un plazo de tiempo determinado, ejemplo 80% en 20 segundos.
- Resolución en la primera solicitud. Porcentaje de solicitudes recibidas que pudieron ser resueltas sin necesidad de una segunda solicitud.
- Tiempo de respuesta. Tiempo utilizado para completar una tarea determinada.

Es así que los SLA en general representan para cualquier línea de negocio el reflejo claro y conciso de dos elementos; por un lado, describen los servicios que se cubren bajo el contrato del SLA y además el nivel operativo estándar o normal.

Comparativa con una línea base

Por tanto, los SLA, en general, representan para cualquier línea de negocio un reflejo claro y conciso de dos elementos esenciales: por un lado, describen los servicios que se incluyen bajo el contrato del SLA y, por otro, el nivel operativo estándar o habitual.

Este procedimiento exige llevar a cabo un registro completamente detallado y documentado de las métricas de rendimiento del sistema de manera general, abarcando todos los módulos necesarios y recolectando muestras de los parámetros que se consideren importantes o adecuados para validar la eficiencia y efectividad del sistema, y/o para servir como referencia en el futuro.

Este tipo de pruebas permite asegurar que el rendimiento del sistema no se degrade con el paso del tiempo, soportando cambios que, en principio, buscan mejorar la escalabilidad y el tamaño del sistema. Sin embargo, en el evento de que las medidas base se deterioren, es posible identificar los puntos críticos o fallas para implementar

medidas correctivas que restablezcan el sistema a los parámetros de la línea base. Esto garantiza que la experiencia del usuario no se vea alterada y, por el contrario, perciba una mejora sustancial y continua en la aplicación.

1.1. Métricas de evaluación

Durante el procedimiento de evaluación de la escalabilidad de un sistema, es esencial tener en cuenta ciertos parámetros cuantificables clave para caracterizar su rendimiento, como el tiempo de respuesta, la latencia, la utilización de recursos, entre otros. Estos parámetros facilitan la estimación de la capacidad del sistema para incrementar el número de usuarios y/o solicitudes sin comprometer la calidad del servicio ofrecido.

A continuación, se presentan algunas métricas de evaluación para facilitar su identificación:

- **Tiempo de respuesta**. Tiempo transcurrido entre el inicio de una solicitud o comando a un sistema informático y el principio de la respuesta del sistema (IEEE, 2000).
- **Latencia**. Tiempo entre la emisión de una solicitud y el comienzo de procesos reales en el servidor de esa petición (cuando inicia la transferencia de datos).
- **“Throughput”**. Según Meier et al. (2007) se refiere al número de unidades de trabajo que se puede manejar por unidad de tiempo (rendimiento) por ejemplo, las solicitudes por segundo, las solicitudes por día, visitas por segundo informe, por año, etc.

- **Utilización de recursos**. Métrica de un consumo del recurso tecnológico como el procesador, la memoria, el disco E/S, y la red I/O.
- **Tiempo de carga**. Tiempo que transcurre entre la solicitud y la respuesta de lectura de información que se busca dentro de los componentes de almacenamiento, ya sea de largo plazo como el disco duro o de corto plazo como la memoria RAM. También es posible aplicarlo a los procesos de asignación de memoria inicial, la cual es reservada para ser usada durante un proceso por el programa (IEEE, 2000).
- **Tasa de errores**. Meier y otros (2007) indican que es la relación entre el número de elementos erróneos que pueden ser bits, datos, código o bloques, en una señal digital recibida y el número total de elementos emitidos durante un intervalo de tiempo determinado.
- **Usuarios simultáneos.** El término se refiere a la característica de un sistema de proveer un servicio y/o procesamiento a múltiples usuarios conectados al mismo instante de tiempo.
- **Solicitudes por segundo**. Tasa de peticiones que puede soportar un sistema en función del tiempo (IEEE, 2000).
- **Transacciones satisfactorias/fallidas**. Diferencia entre el número de peticiones al sistema que terminaron con una respuesta apropiada frente al mismo número de peticiones pero que terminaron con respuesta errada.
- **Utilización de la CPU**. Nivel de ocupación (carga) y el tiempo de procesamiento de datos que le toma al equipo “hardware” la gestión de la solicitud.

- **Utilización de la memoria**. Nivel de ocupación (carga) y el tiempo de procesamiento de datos que le toma a la memoria de corta duración RAM la gestión de la solicitud.
- **E/S de disco**. Diferencia entre el número de procesos que realizan un evento de entrada y uno de salida en el disco duro del componente.

Una prueba de rendimiento tiene como objetivo evaluar el comportamiento de la aplicación en un entorno de producción cuando se enfrenta a una gran demanda de uso (grandes cantidades de peticiones y transacciones). Existe una amplia variedad de pruebas de este tipo; no obstante, solo se explorarán las siguientes:

Video 1. Pruebas de rendimiento

Enlace de reproducción del video

Síntesis del video: pruebas de rendimiento

Las pruebas de rendimiento son evaluaciones esenciales que determinan la respuesta de un sistema o aplicación bajo distintas condiciones de carga. A continuación, exploraremos las principales pruebas de rendimiento empleadas para evaluar sistemas y aplicaciones que incluyen:

- **Pruebas de carga.** Evalúan como el sistema maneja diferentes volúmenes de trabajo como, por ejemplo, verificar si puede soportar 1000 usuarios concurrentes. Inicialmente se compara el rendimiento con una base establecida con pocos o ningún usuario concurrente para luego incrementar gradualmente la carga y observar la respuesta del sistema.
- **Pruebas de estrés**. Determinan la capacidad del sistema bajo condiciones extremas como un número máximo de usuario o volumen de datos. El objetivo es identificar el punto de saturación donde el sistema ya no puede procesar transacciones de manera eficiente.
- **Pruebas de resistencia**. Verifica como el sistema se comporta bajo una carga constante y moderada durante un periodo prolongado. Esto ayuda a detectar problemas como fugas de memoria o conexiones sin cerrar que pueden surgir solo después de un uso continuado.
- **Pruebas de estabilidad.** Se centran en la capacidad del sistema para manejar una carga esperada de manera continua. Ayudando a identificar problemas de optimización y fugaz de memoria.

Cada una de estas pruebas es crucial para asegurar que un sistema o aplicación funcione correctamente bajo diversas condiciones de uso, garantizando una

experiencia de usuario optima y la estabilidad del sistema en el entorno de producción.

1.2. Niveles de carga

Los niveles de carga para una prueba dependen en gran medida de la estrategia planificada, la cual se fundamentará en el estado actual de la aplicación. Esto implica un análisis de la infraestructura, la tecnología utilizada, entre otros aspectos, así como el conocimiento de los problemas existentes, los módulos que realmente requieren validación y los objetivos que se pretenden alcanzar a futuro con la aplicación.

Con este propósito, resulta crucial recopilar información de las distintas áreas de la empresa que interactúan con la aplicación, así como de los propios clientes finales. Su perspectiva y percepción sobre el producto ofrecido pueden influir significativamente en la modificación y refinamiento de los objetivos de las pruebas y, por consiguiente, en los niveles de carga.

De esta manera, los niveles de carga varían acorde a los tipos de prueba, de la siguiente manera:

- **Aumento progresivo de carga**. Inicia con un número bajo de usuarios, peticiones y/o transacciones; puede ser en un porcentaje de 10% de la capacidad que se cree soporta o se aspira para la aplicación y se va aumentando en pasos de valores enteros (al 20%, al 40%, etc.) hasta llegar al 100% de la capacidad que alcance la aplicación. Tomando muestras en cada nuevo paso de la respuesta observada.

- **Puntos de inflexión y sus límites**. Este tipo de nivel de carga depende del anterior; dado que, es necesario comprender de antemano el punto crítico de falla de la aplicación, para llevar el "test" a valores por encima o por debajo durante un tiempo determinado que debe ir variando dependiendo de la estrategia que se desee analizar (horas pico de la aplicación, nivel más alto de carga, horas de la noche, entre otros).
- **Aumento progresivo de carga en el tiempo**. De la misma forma que el anterior modo, pero manteniendo el proceso por más tiempo a fin de simular un flujo constante de peticiones y/o transacciones.
- **Aumento y disminución abrupta de la carga**. Este método consiste en lanzar diferentes pruebas con carga muy alta que llegue a niveles cercanos al 100% y luego cargas muy bajas del orden del 10%, para analizar cómo responde el sistema después de un pico tan alto, o cuánto le toma al sistema recuperarse si hubo fallas.

También es crucial revisar los tipos de fallas que el equipo de pruebas debe estudiar, quienes se esfuerzan constantemente por identificar problemas en los sistemas que evalúan, buscando amenazas potenciales o procesos que puedan desencadenar resultados indeseados.

En el ámbito de las pruebas de rendimiento, hay fallos comunes que ya están plenamente identificados y que es esencial detectar con facilidad para, por ende, resolverlos de la mejor manera posible. A continuación, se listan los siguientes:

- **Cuellos de botella**. En sistemas, los recursos “hardware”, “software” y la red pueden limitar el flujo de información y velocidad de transmisión, creando posibles cuellos de botella. Estos errores permiten a los “testers” identificar componentes problemáticos durante pruebas de carga y estrés, afectando el rendimiento y escalabilidad al limitar el procesamiento de información y conexiones de usuarios simultáneos.
- **Procesos zombies**. En el ámbito del desarrollo de “software”, las transacciones y eventos están vinculados a procesos en el “hardware”, interconectados por la complejidad de las aplicaciones multifuncionales. Esta interacción crea una estructura de árbol donde los procesos están interrelacionados, pero cuando un proceso hijo finaliza y el padre no recicla correctamente este proceso finalizado, surge un “proceso zombie”. Las pruebas de resistencia son clave para detectar estos errores.
- **Conexiones sin cerrar o colgadas**. Al igual que los procesos zombies, las conexiones sin cerrar ocurren por eventos disparados desde la aplicación hacia el servidor de base de datos, resultado de transacciones que no confirman su cierre. Esto provoca que la conexión quede bloqueada, impidiendo su uso por otros eventos o procesos hasta que la base de datos deja de responder a las solicitudes de la aplicación.
- **Fugas de memoria**. Al ejecutar eventos en una aplicación, se solicitan recursos que llevan a que el “hardware” reserve bloques de memoria RAM de corto plazo. Una vez concluido el evento, se espera que este bloque de memoria se libere para su reutilización. Sin embargo, si el sistema no logra liberar correctamente estos bloques de memoria, se generan espacios bloqueados

pero inutilizados, lo que eventualmente puede llevar a la imposibilidad de solicitar más memoria por estar aparentemente toda ocupada.

- **Fallos de configuración**. Este tipo de error surge del control que el programador del sistema ejerce, ya que es este quien modifica los parámetros predeterminados en los componentes de “software” y “hardware” para realizar configuraciones personalizadas de estas herramientas.

Las métricas mencionadas, los tipos de pruebas, los niveles de carga, y los tipos de fallas son esenciales en el proceso de evaluación de la escalabilidad de un sistema de “software”. Resulta crucial definir una hoja de ruta para las pruebas, lo que implica la necesidad de identificar el tipo de pruebas a ejecutar para establecer un plan basado en una metodología que evalúe la escalabilidad de un sistema de “software” mediante el uso de una herramienta que facilite el abordaje de estos parámetros.

2. Instalación y configuración de herramientas de pruebas de rendimiento

Una vez revisados y comprendidos los conceptos acerca de la temática de la evaluación de escalabilidad, es posible profundizar en los pasos requeridos para la instalación y configuración de una herramienta de pruebas de rendimiento.

Para lograr que el proceso sea lo más satisfactorio posible, se necesita instalar algunas herramientas que se encuentran en internet, las cuales pueden ser gratuitas o "premium" (pagas), con lo cual es posible realizar cualquier tipo de prueba, funcionales, rendimiento o regresión, entre otras. Es importante tener presente que estas herramientas son imprescindibles en el negocio del "software", en equipos de alto rendimiento.

- **JMETER**. Apache JMeter es una destacada herramienta de código abierto para pruebas de rendimiento y estrés de aplicaciones web, desarrollada por la fundación Apache "Software". Permite generar alta carga para facilitar el análisis de rendimiento y está disponible para descarga gratuita en su página oficial.
- **HP LoadRunner**. Hewlett-Packard ofrece una alternativa para pruebas de rendimiento y estrés con dos versiones: una de pago en modo SaaS y una gratuita con funcionalidades limitadas en su edición "Community".
- **OctoPerf**. Esta herramienta, basada en Apache JMeter, es de pago y destaca por no requerir instalación. Permite crear y monitorear escenarios, ejecutar pruebas y analizar resultados desde un único punto.
- **LoadNinja**. La herramienta de SmartBear facilita la realización y ejecución rápida de pruebas de carga, ayudando en el diagnóstico de problemas de rendimiento

en sistemas de “software”. Ofrece datos precisos, legibles y accionables sin requerir extensas líneas de código.

- **WebLOAD.** RadView ofrece una alternativa en herramientas de pruebas de rendimiento que destaca por su integridad y escalabilidad. Facilita la simulación de miles de usuarios de manera concurrente o simultánea para identificar puntos de inflexión y cuellos de botella.

Enseguida, realizaremos un ejercicio de instalación. Se ha seleccionado JMeter como herramienta de pruebas debido a que es gratuita, de acceso rápido e intuitiva en su uso. No obstante, es necesario preparar un ambiente web para realizar las pruebas, por lo que se necesita instalar Java.

Según Torres (2020) Java Development Kit es un “software” de Oracle que suministra una serie de herramientas de desarrollo para crear aplicaciones en lenguaje de programación Java. Este puede instalarse de manera directa en equipos locales o en servidores en línea y unidades de red, permitiendo la distribución y el trabajo cooperativo.

Como se venía comentando, la aplicación gratuita JMeter ofrece la posibilidad de ejecutar un plan de pruebas de rendimiento de manera fácil y rápida. Gracias a la usabilidad de esta herramienta y el conocimiento obtenido sobre las distintas métricas y valores de carga, se podrá definir una metodología para establecer el plan de pruebas adecuado según el objetivo de la aplicación.

A continuación, le invitamos a observar un ejercicio práctico en el que se identifican los pasos de descarga de JMeter y la manera de realizar un plan de pruebas:

Video 2. Instalar JMeter

Enlace de reproducción del video

Síntesis del video: instala JMeter
El video presenta una introducción a la herramienta JMeter, que permite realizar pruebas de carga, estrés y estabilidad de manera gratuita, ya que es de código abierto. Se explica cómo descargar e instalar Java “Development Kit” (JDK) como requisito previo, así como la descarga e instalación de JMeter desde su página oficial. Se destacan las diferentes funcionalidades de la herramienta, como la creación de planes de prueba para aplicaciones web y no web, y la obtención de informes de rendimiento. Se proporcionan instrucciones detalladas sobre cómo descomprimir y ejecutar la herramienta una vez descargada. El video concluye con la presentación de

la interfaz de usuario de JMeter y la iniciación del proceso de diseño de una prueba de carga como ejemplo exploratorio del uso de la herramienta.

Video 3. Caso de prueba

Enlace de reproducción del video

Síntesis del video: caso de prueba
El video muestra una simulación de una aplicación de reserva de vuelos para fines didácticos. Explica el proceso paso a paso, desde la selección de un vuelo hasta la confirmación de la compra. Se enfatiza que la aplicación no tiene validaciones complejas, ya que su objetivo es educativo. Se guía al espectador a través de la selección de vuelos, ingreso de datos y confirmación de compra. Se menciona la importancia de seleccionar un proceso de negocio para la prueba, como la radicación

de un documento en un sistema de gestión documental, para evaluar el rendimiento del servidor o la aplicación.

Video 4. Primer plan de pruebas con interfaz JMeter

Enlace de reproducción del video

Síntesis del video: primer plan de pruebas con interfaz JMeter
El video guía al espectador a través del proceso de creación de un primer plan de pruebas utilizando la interfaz de JMeter. Se explica cómo abrir la interfaz y asignar un nombre al plan de pruebas. Se introduce el concepto de grupos de hilos, explicando que representan procesos simultáneos dentro de la aplicación. Se ilustra

cómo configurar el número de hilos y la frecuencia de ejecución para simular la carga de trabajo. Se destaca la importancia de comprender el comportamiento de la herramienta para luego automatizar el proceso de pruebas. Finalmente, se menciona la necesidad de aprender sobre “proxies” para que la herramienta pueda observar y replicar los procesos de manera efectiva.

Video 5. Identificar el proxy

Enlace de reproducción del video

Síntesis del video: identificar el proxy
El video explica el concepto de un servidor Proxy y su papel en la gestión y control del acceso a Internet en redes corporativas. Se menciona que, aunque su uso ha disminuido con la evolución de los dispositivos de firewall, aún es utilizado por

algunas empresas debido a su economía y capacidades de control. Se destaca que todos los navegadores y sistemas pueden configurarse para utilizar un Proxy. Luego, se muestra cómo Apache, JMeter utiliza este protocolo para aprender y replicar conexiones a través de un Proxy. Se guía al espectador a través de la configuración de un servidor Proxy en JMeter, resaltando la importancia de asignar el grupo de hilos correspondiente para permitir su funcionamiento adecuado.

Video 6. Correr el proxy

Enlace de reproducción del video

Síntesis del video: correr el proxy
El video guía al espectador a través del proceso de configuración y uso de un servidor Proxy en un navegador web. Se explica cómo habilitar y correr un Proxy en la herramienta, así como la configuración del navegador para que navegue a través del

mismo. Adicionalmente se muestra cómo ajustar la configuración en diferentes navegadores, como Chrome y Firefox, así como aceptar los certificados de seguridad para permitir la navegación a través del Proxy. Se destaca la importancia de deshabilitar el Proxy una vez finalizadas las pruebas para evitar problemas de navegación posteriormente.

Video 7. Peticiones en JMeter

Enlace de reproducción del video

Síntesis del video: peticiones en JMeter
En el video, se muestra cómo se ejecuta y captura el comportamiento de un Proxy en JMeter mientras se navega por un sitio web. Se observa cómo se registran las solicitudes realizadas durante la navegación, desde la página de inicio hasta la confirmación de una compra. Se destaca la importancia de detener el Proxy una vez

finalizadas las pruebas para restaurar la navegación normal. Se destaca también la funcionalidad de JMeter para grabar y reproducir solicitudes, lo que permite simular procesos de manera concurrente.

Video 8. Guardar el "script" y reconfigurar

Enlace de reproducción del video

Síntesis del video: guardar el "script" y reconfigurar
En el video, se muestra cómo guardar un "Script" en JMeter y reconfigurar el proceso utilizando prefijos para identificar las transacciones. Se resalta la importancia de nombrar adecuadamente las transacciones para facilitar la comprensión y organización del "Script". También se realiza una demostración práctica de cómo utilizar prefijos para asignar nombres a las transacciones y mejorar la claridad del "Script".

Video 9. Árbol de resultados para ejecutar hilos

Enlace de reproducción del video

Síntesis del video: árbol de resultados para ejecutar hilos
El video muestra cómo ejecutar un grupo de hilos en JMeter y añadir el receptor "Árbol de resultados" para visualizar la ejecución de los hilos. Se demuestra cómo identificar errores en las solicitudes mediante el análisis del árbol de resultados, destacando que las solicitudes exitosas se muestran en verde mientras que las fallidas muestran mensajes de error. Se explica cómo ejecutar múltiples veces el proceso y la configuración para enviar solicitudes concurrentemente, con ejemplos de intervalos de tiempo entre cada envío. Finalmente, se limpian los resultados anteriores antes de ejecutar el proceso nuevamente.

Video 10. Receptor de informe agregadolar

Enlace de reproducción del video

Síntesis del video: receptor de informe agregadolar
El video introduce el receptor de informe agregado en JMeter, que proporciona métricas importantes como el tiempo máximo y mínimo de respuesta de las solicitudes, el porcentaje de tiempo en línea, los errores, el rendimiento, el número de solicitudes por hora y la cantidad de kilobytes transferidos por segundo. Se explica cómo acceder a este receptor mediante la opción "Añadir receptor" en el grupo de hilos. Al ejecutar el proceso, el receptor de informe agregado muestra estadísticas detalladas, como la demora más larga y la capacidad de respuesta del servidor bajo carga. Se destaca la importancia de estas métricas para evaluar el rendimiento y la capacidad del servidor. Además, se menciona que existen otras herramientas en JMeter que se explorarán más adelante.

Video 11. Simular comportamientos del "script"

Enlace de reproducción del video

Síntesis del video: simular comportamientos del "script"
El video concluye explicando cómo simular comportamientos más naturales en los "scripts" de JMeter. Se introduce el concepto de temporizador para agregar retrasos entre solicitudes y se muestra cómo usar el temporizador constante y el temporizador aleatorio uniforme. Además, se discute la importancia de estas herramientas para simular el comportamiento humano al interactuar con una aplicación. Se menciona brevemente la posibilidad de realizar pruebas de estrés y pruebas de carga más densas utilizando múltiples clientes JMeter configurados en una red. Se destaca la abundante documentación disponible para explorar en mayor profundidad el uso de JMeter y cómo diseñar pruebas más elaboradas. Finalmente, se invita a los espectadores a seguir aprendiendo sobre esta herramienta y su aplicación en pruebas de rendimiento y carga.

3. Metodología para la evaluación de escalabilidad

Las pruebas de rendimiento requieren de una estrategia bien planificada para asegurar que los objetivos iniciales y los resultados esperados coincidan con lo deseado por todo el equipo de trabajo. El planteamiento metodológico siguiente es el proceso más importante, ya que define el alcance de dichas pruebas de rendimiento, la directriz del nivel de carga y los objetivos de nivel de servicio (SLO). Es decir, determina las partes o módulos que serán objeto de estudio y los acuerdos de nivel de servicio (SLA), así como la expectativa de calidad en el servicio prestado que se espera reciba el cliente final. Por lo tanto, siempre que se necesite elaborar una metodología para la evaluación, se deben adoptar las cinco etapas mencionadas a continuación:

- Planificación de las pruebas.
- Evaluación del entorno de pruebas.
- Selección de métricas y niveles de carga.
- Definición casos de prueba.
- Ejecución de las pruebas.

Le invitamos a revisar el siguiente contenido para conocer los elementos que se deben tener en cuenta en cada una de estas etapas:

a) **Planificación de las pruebas**. En esta etapa inicial, el equipo debe identificar las razones para realizar pruebas de rendimiento, como cambios tecnológicos o expansión de clientes, mediante un análisis detallado del estado actual de la aplicación. Esto permitirá establecer objetivos claros y alcanzables. Es crucial determinar qué componentes de la aplicación modular necesitan ser probados y cuáles no, para asegurar una evaluación efectiva.

b) **Evaluación del entorno de pruebas**. Esta etapa, se enfoca en identificar detalladamente las características, herramientas, tecnologías y versiones del aplicativo para elegir correctamente el tipo de pruebas de rendimiento y la herramienta más adecuada, considerando costos, esfuerzo y tiempo.

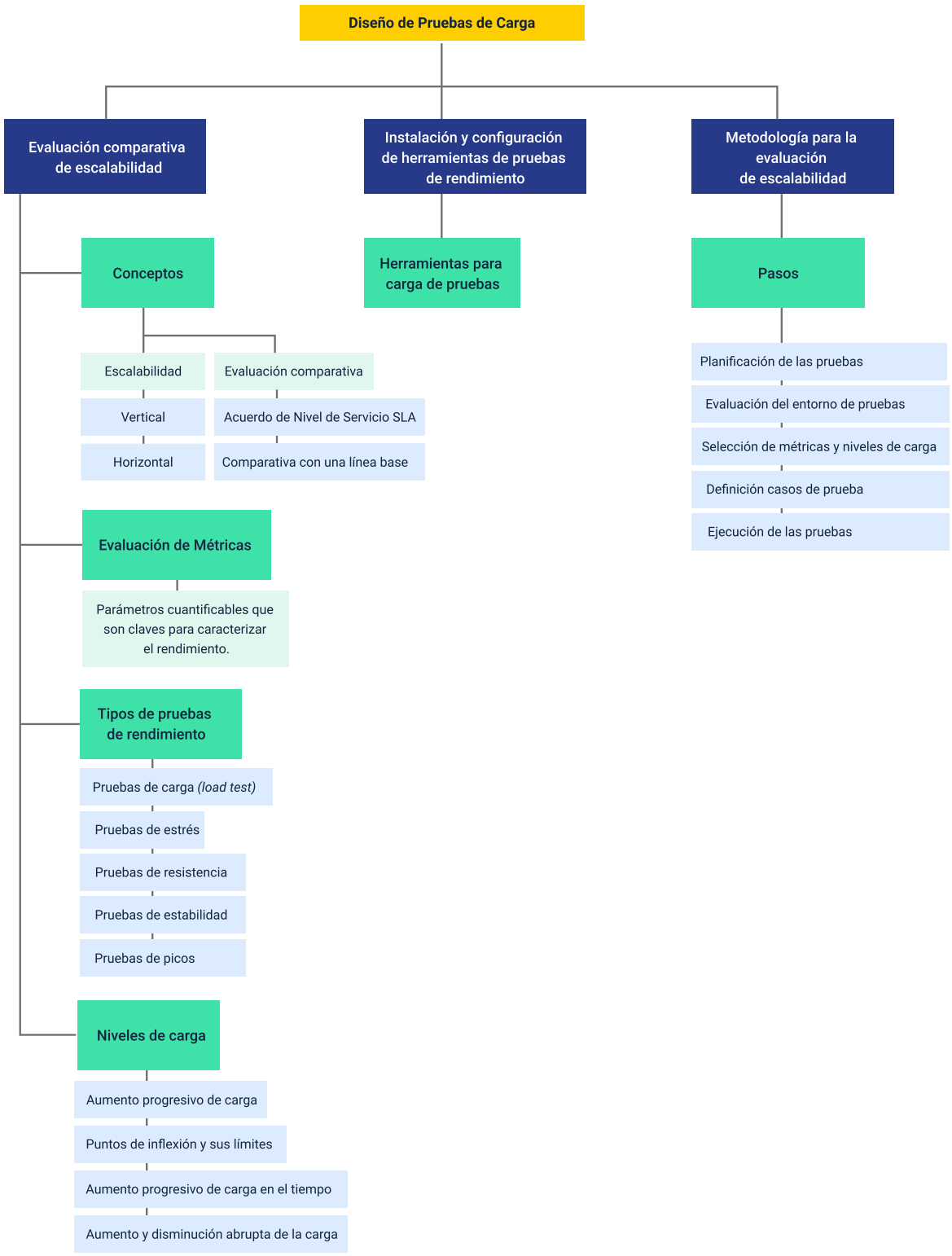
c) **Selección de métricas y niveles de carga**. Esta etapa del proceso implica estrecha conexión con etapas anteriores, enfocándose en la selección de parámetros para pruebas de rendimiento basada en la asesoría de expertos y en objetivos claros establecidos inicialmente, con el fin de identificar las causas principales de problemas detectados.

d) **Definición casos de pruebas**. Esta etapa, la más analítica, involucra definir la carga de trabajo esperada como referencia (número de conexiones, solicitudes, etc.), y establecer objetivos de rendimiento basados en la percepción del usuario, que pueden concretarse en acuerdos de nivel de servicio. También requiere seleccionar los subsistemas a testear, considerando la variedad de procesos de negocio y estimar el flujo de negocio para las pruebas. Adicionalmente, se puede realizar un "benchmark", comparando la carga esperada en la aplicación con "software" similar en tecnología y condiciones.

e) **Ejecución de las pruebas**. En la etapa final, se llevan a cabo y gestionan las pruebas, registrando los resultados para elaborar informes, gráficas, tablas comparativas y matrices de correlación que faciliten la interpretación de los datos de forma clara y eficiente para el equipo. Luego, se ajustan las fallas detectadas y se repiten las pruebas para alcanzar los objetivos establecidos al principio de la metodología.

En esta sección, se han unificado los conocimientos y herramientas para desarrollar una metodología que permita completar un plan de pruebas de rendimiento. Este plan se basa en cinco procesos: la planificación, la evaluación del entorno, la selección de métricas y niveles, la definición de casos y la ejecución de las pruebas. El propósito de estos procesos es cumplir con los objetivos trazados para realizar mejoras o ajustes relacionados con el rendimiento de la aplicación.

Síntesis


Diseño de Pruebas de Carga
Evaluación comparativa de escalabilidad
Instalación y configuración de herramientas de pruebas de rendimiento
Metodología para la evaluación de escalabilidad
Conceptos
Herramientas para carga de pruebas
Pasos
Escalabilidad
Vertical
Horizontal
Evaluación comparativa
Acuerdo de Nivel de Servicio SLA
Comparativa con una línea base
Planificación de las pruebas
Evaluación del entorno de pruebas
Selección de métricas y niveles de carga
Definición casos de prueba
Ejecución de las pruebas
Evaluación de Métricas
Parámetros cuantificables que son claves para caracterizar el rendimiento.
Tipos de pruebas de rendimiento
Pruebas de carga (load test)
Pruebas de estrés
Pruebas de resistencia
Pruebas de estabilidad
Pruebas de picos
Niveles de carga
Aumento progresivo de carga
Puntos de inflexión y sus límites
Aumento progresivo de carga en el tiempo
Aumento y disminución abrupta de la carga

Material complementario

Tema	Referencia	Tipo de material	Enlace del recurso
2. Instalación y configuración de herramientas de pruebas de rendimiento.	Oracle. (s.f.). Descargas de Java.	Web	https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html
2. Instalación y configuración de herramientas de pruebas de rendimiento.	Eruiz. (2022). Instalar Java JDK windows 10. [Video]. YouTube.	Video	https://www.youtube.com/watch?v=xr069CmHWcs
2. Instalación y configuración de herramientas de pruebas de rendimiento.	Apache JMeter. (2021).	Instalador (Web Oficial)	https://JMeter.apache.org/
2. Instalación y configuración de herramientas de pruebas de rendimiento.	Apache Maven. (2021).	Instalador (Web Oficial)	https://maven.apache.org/
2. Instalación y configuración de herramientas de pruebas de rendimiento.	Bravo, N. (2015). Pruebas de Carga & Stress: JMeter. [Vídeo]. YouTube.	Video	https://www.youtube.com/watch?v=CdH9Y3MQ33g

Glosario

Acuerdos de nivel de servicios: es un acuerdo escrito entre un proveedor de servicio y su cliente con objeto de fijar el nivel acordado para la calidad de dicho servicio.

Api: la interfaz de programación de aplicaciones, es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizada por otro "software" como una capa de abstracción.

"Cluster": se aplica a los conjuntos de nodos construidos mediante la utilización de "hardware" comunes y que se comportan como si fuesen una única computadora.

Escalabilidad: es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para reaccionar y adaptarse sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

IDE: entorno de desarrollo integrado, es una aplicación informática que proporciona servicios integrales para facilitarle al programador el desarrollo de "software".

"Key Performance Indicators": indicador clave o medidor de desempeño o indicador clave de rendimiento, es una medida del nivel del rendimiento de un proceso.

Línea base: según el estándar de la IEEE es una especificación o producto que ha sido revisado formalmente, sobre el que se ha llegado a un acuerdo, y que de ahí en adelante servirá como base para un desarrollo posterior que puede cambiarse solamente a través de procedimientos formales de control de cambios.

“Memory leaks”: una fuga de memoria es un error de “software” que ocurre cuando un bloque de memoria reservada no es liberado en una transacción.

Mesa de ayuda: es un conjunto de recursos tecnológicos y humanos, para prestar servicios con la posibilidad de gestionar y solucionar todas las posibles incidencias de manera integral, junto con la atención de requerimientos relacionados con las Tecnologías de la Información y la Comunicación (TIC).

“Offline”: indica que no se tiene conexión a la red de internet.

“Rampup”: en informática se puede describir como el aumento exponencial e imprevisto en la carga de peticiones o transacciones en un sistema.

“Slot”: ranura de expansiones un conector o puerto de expansión en la placa base de la computadora.

“Stakeholder”: el interesado, parte interesada o involucrado es una persona, organización o empresa que tiene interés en una empresa u organización dada.

“Throughput”: la tasa de transferencia efectiva es el volumen de trabajo o de información neto que fluye a través de un sistema, como puede ser un sistema “software”.

Referencias bibliográficas

IEEE. (2000). IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. https://ieeexplore.ieee.org/document/875998

Mazano, J. l. (2017). Informes SLA, evita problemas con Pandora FMS.

Meier, J.D.,et al. (2007). Performance Testing Guidance for Web Applications.

Oracle. (2010). Introduction to Performance Monitoring. s.l.: Oracle, 2010. Tomada de las presentaciones del curso Oracle University de administración de weblogic.

Torres. A. (2020). Instalar JDK java en Windows 10.

Créditos

Nombre	Cargo	Regional y Centro de Formación
Milady Tatiana Villamil Castellanos	Responsable del Ecosistema	Dirección General
Olga Constanza Bermúdez Jaimes	Responsable de Línea de Producción	Centro de Servicios de Salud - Regional Antioquia
Peter Emerson Pinchao Solis	Experto Temático	Centro de Teleinformática y Producción Industrial - Regional Cauca
Danny Alejandro Solano Concha	Evaluadora Instruccional	Centro de Teleinformática y Producción Industrial - Regional Cauca
Paola Alexandra Moya	Evaluadora Instruccional	Centro de Servicios de Salud - Regional Antioquia
Carlos Julián Ramírez Benítez	Diseñador de Contenidos Digitales	Centro de Servicios de Salud - Regional Antioquia
Jhon Jairo Urueta Alvarez	Desarrollador Fullstack	Centro de Servicios de Salud - Regional Antioquia
Edgar Mauricio Cortés García	Actividad Didáctica	Centro de Servicios de Salud - Regional Antioquia
Daniela Muñoz Bedoya	Animador y Productor Multimedia	Centro de Servicios de Salud - Regional Antioquia
Laura Gisselle Murcia Pardo	Animador y Productor Multimedia	Centro de Servicios de Salud - Regional Antioquia
Andrés Felipe Guevara Ariza	Locución	Centro de Servicios de Salud - Regional Antioquia
Margarita Marcela Medrano Gómez	Evaluador para Contenidos Inclusivos y Accesibles	Centro de Servicios de Salud - Regional Antioquia

Nombre	Cargo	Regional y Centro de Formación
Daniel Ricardo Mutis Gómez	Evaluador para Contenidos Inclusivos y Accesibles	Centro de Servicios de Salud - Regional Antioquia
Luis Gabriel Urueta Álvarez	Validador de Recursos Educativos Digitales	Centro de Servicios de Salud - Regional Antioquia
Jaime Hernán Tejada Llano	Validador de Recursos Educativos Digitales	Centro de Servicios de Salud - Regional Antioquia