ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

EVALUACIÓN DE HERRAMIENTAS OPEN SOURCE PARA PRUEBAS DE FIABILIDAD Y RENDIMIENTO DE APLICACIONES WEB

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

EDISON FERNANDO MAILA MAILA
fernando_efml@hotmail.com

DIRECTOR: MSc. Monserrate Intriago
Monserrate.intriago@epn.edu.ec

Quito, diciembre 2017

DECLARACIÓN

Yo, Edison Fernando Maila Maila, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Fernando Maila

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Edison Fernando Maila Maila, bajo mi supervisión.

Msc. Monserrate Intriago
DIRECTOR DE PROYECTO

AGRADECIMIENTO

Gracias a mi familia que han sido el apoyo fundamental en este proceso de formación profesional, en especial a mi hermano Luis por la confianza que tiene depositada en mí, a mi madre Olivia y mi abuelita Carmen, quienes siempre me han impulsado a cumplir mis metas.

Gracias a todos los profesores que tuve en todo el transcurso de la carrera, a mi tutora de Proyecto Integrador MSc. Monserrete Intrigado.

A una persona que en este último año se ha convertido en una alguien muy importante en mi vida y estuvo animándome para lograr este objetivo. A mis amigos que siempre estuvieron preocupados del avance de mi vida universitaria dándome palabras de aliento.

Fernando

DEDICATORIA

A mi hermano, mi madre y mi abuelita, quienes siempre me han apoyado.

CONTENIDO

Índice de figuras

Índice de Tablas

RESUMEN

El desarrollo de este proyecto inicia con la búsqueda en la red de herramientas Open Source que permitan la evaluación de métricas de fiabilidad y/o rendimiento, posteriormente se seleccionó una herramienta la cual se ejecuta en una aplicación web.

El objetivo de este proyecto integrador es evaluar herramientas Open Source para realizar pruebas de fiabilidad y rendimiento de aplicaciones web. Este proyecto está enfocado a los las micro, pequeños y medianos desarrolladores, quienes no pueden acceder a herramientas de evaluación de calidad con licencia propietaria, por lo tanto, este proyecto pretende demostrar que existen herramientas Open Source que funcionan adecuadamente y que está al alcance de todos.

Para llegar al cumplimiento del objetivo realiza un proceso que está conformado de varias fases: revisión literaria, estrategia, métodos para genera datos y el análisis de los datos.

Los resultados obtenidos en una primera búsqueda son dieciocho herramientas, posteriormente se filtran bajo ciertos parámetros que reducen la lista de herramientas a ocho. Finalmente, luego de someter las herramientas a pruebas se puede comprobar un funcionamiento satisfactorio.

Las herramientas Open Source para evaluar rendimiento de aplicaciones web son una buena alternativa frente al software propietario, cumplen un adecuado funcionamiento. Sin embargo, cabe aclarar que a un nivel de exigencia elevado no funcionaría bien.

Palabras clave: Open Source, calidad de software, rendimiento, fiabilidad, evaluación.

ABSTRACT

This project begins with the search in Internet of Open Source tools that allows the evaluation of reliability and performance metrics. Then a tool is selected and it is executed to access.

The objective of this integrative project is to find Open Source tools to perform reliability and performance tests of web applications. This project is focused on microenterprises, small and medium developers, who can not access quality assessment tools with proprietary licenses, therefore, this project aims to demonstrate that there is Open Source that works properly and is within reach of All Everyone.

To achieve the objective, perform a process that consists of several phases: literary review, strategy, methods for personal data and data analysis.

The results obtained in a first search are tools of eighteen, then filtered under parameters that reduce the list of tools to eight. Finally, after sending the tools to tests you can verify a satisfactory operation.

Open source tools to evaluate web application performance are a good alternative to proprietary software, they comply with an adequate functioning. However, it should be clarified that at the level of high demand it would not work well.

Keywords: Open Source, software quality, performance, reliability, evaluation.

1. INTRODUCCIÓN

Los MIPYMES (micro, pequeñas y medianas empresas) buscan innovar en sus procesos con el fin de optimizar recursos, seguridad en el manejo de información, ahorrar dinero y ganar tiempo. En la búsqueda de cumplir esos objetivos se apoyan en productos de software. Según la revista Vanguardia los negocios que son gestionados con software pueden incrementar sus ganancias hasta un 70% [1].

Si bien, la importancia del software en la actualidad es indiscutible, existe un mercado amplio y variado en cuanto a productos de software. La calidad de un producto de software determinará el éxito en cuanto a la aceptación de los usuarios.

Las grandes empresas de desarrollo de software cuentan con áreas destinadas a realizar pruebas a los productos de software tanto en su etapa de desarrollo como producción, para ello usan herramientas con licencia propietaria. Los pequeños desarrolladores por situación económica posiblemente no pueden acceder a estas herramientas de pago.

Este proyecto busca demostrar el funcionamiento de herramientas Open Source para evaluar la calidad de las aplicaciones web, presentado como una alternativa a las herramientas con licencia propietaria que usan las grandes empresas de desarrollo.

OBJETIVO GENERAL

Evaluar herramientas Open Source para realizar pruebas de fiabilidad y rendimiento de aplicaciones web.

OBJETIVOS ESPECÍFICOS

- Establecer los conceptos relacionados con la planificación y ejecución de las pruebas de software y sus dependencias.
- Identificar 6 herramientas Open Source para pruebas de fiabilidad y rendimiento de aplicaciones web, 3 para cada caso.
- Realizar un estudio comparativo de 3 herramientas identificadas.

- Aplicar una herramienta que mejor se adapte en una pequeña empresa de desarrollo de aplicaciones web.

1.1 MARCO TEÓRICO

1.1.1 Calidad del producto de Software

La palabra calidad está definida por la Real Academia de la Lengua como "*Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor*" [1]. En el ámbito del desarrollo de software, el modelo de calidad ISO/ICE 25000 define la calidad del producto de software como "Capacidad del producto de software para satisfacer las necesidades declaradas e implícitas cuando se utiliza bajo condiciones especificadas" [2]. Al enfocar la definición a la calidad de un producto de software se puede observar que la esencia de calidad sigue siendo la misma, pero con un complemento muy importante que es la satisfacción de necesidades para las que fue hecho el producto de software.

La calidad de un producto de software es una realidad que se la puede percibir en las actividades computacionales diarias y puede representar, sin exagerar, el éxito o el fracaso de un negocio. Un producto de software con fallas puede provocar: pérdidas de tiempo, dinero, ambientales, sociales e incluso la pérdida de vidas humanas. El nivel de gravedad es más alto si los afectados son sistemas críticos.

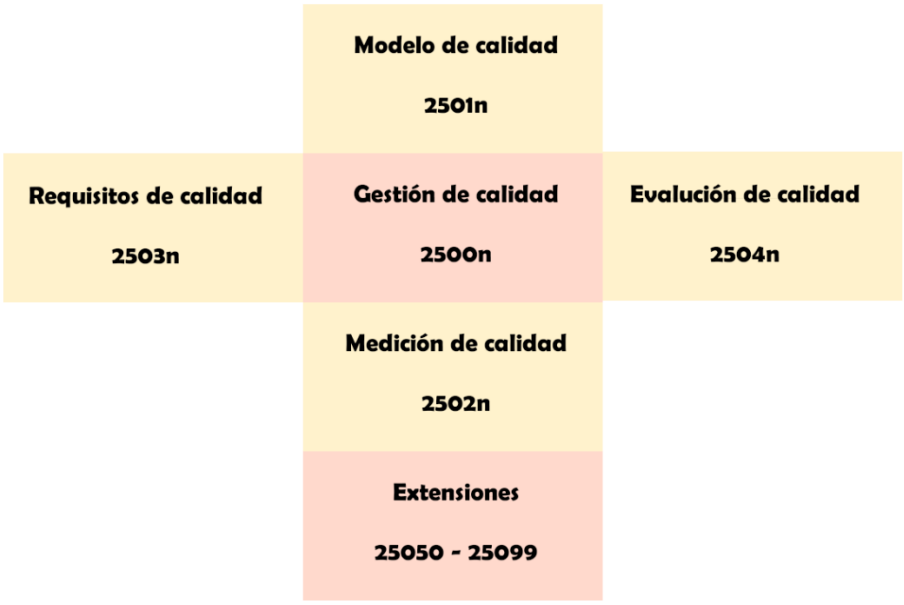
Por otro lado, un producto de software con altos niveles de calidad puede ser un factor clave para el éxito tanto de la empresa desarrolladora como para los clientes que adquieren el software.

1.1.2 Modelo de Calidad ISO/ICE 25010

El modelo de calidad ISO/ICE 25010 es parte de la serie de los procesos de evaluación de la serie *Software Product Quality Requirements and Evaluation* (SQuaRE), este modelo de calidad reemplazó a los estándares ISO/IEC 9126 (Software Product Quality) e ISO/IEC 14598 (Software Product Evaluation) [3].

Esta norma puede ser utilizada junto con otros estándares de la serie SQueRE. En la *Figura 1.1* se puede observar la organización de la serie SQuaRe y su división por familias de estándares.

Figura 1.1 Organización de la serie de normas SQuaRE

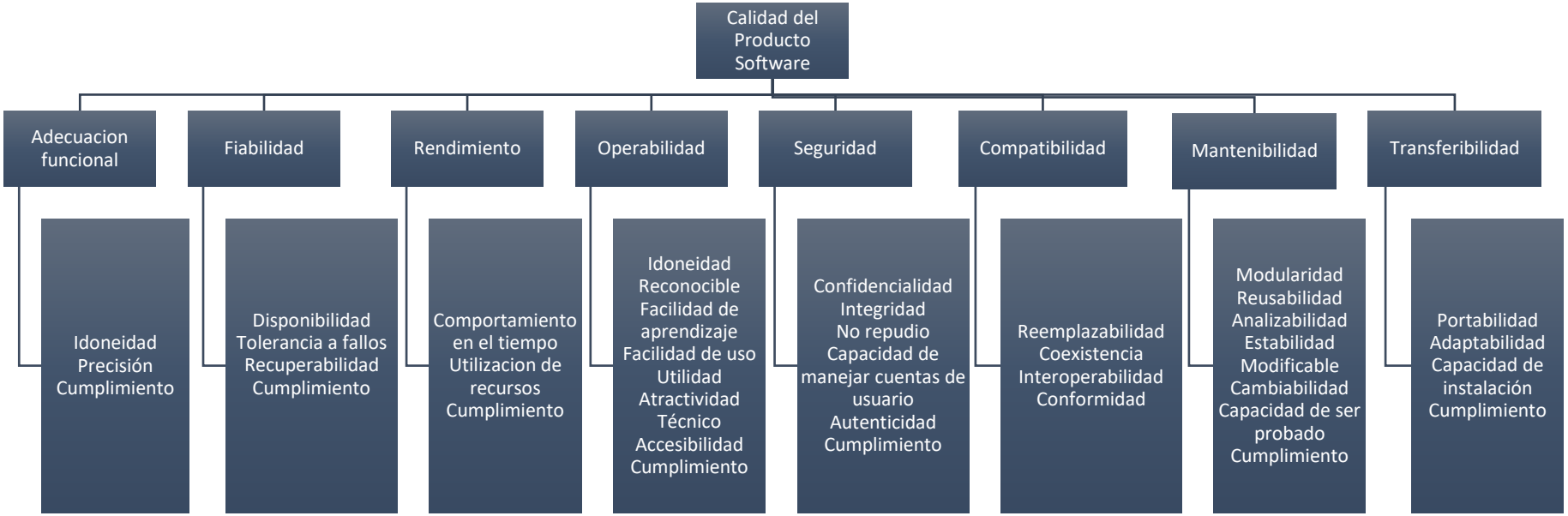


Fuente: ISO/ICE 25010

Autor: ISO/ICE

El modelo ISO/ICE 25010 tiene como propósito establecer un sistema para la evaluación de calidad de productos de software, un producto de software de calidad es el resultado de la calidad de sus elementos, para esto se determina características de calidad como: adecuación funcional, fiabilidad, rendimiento, operabilidad, seguridad, compatibilidad mantenibilidad y transferibilidad. Cada característica tiene subcaracterísticas que se evalúan y determinan la calidad de un producto de software, estas pueden ser medidas internamente como externamente. El modelo ISO/ICE 25010 tienen las siguientes características y subcaracterísticas como se puede visualizar en la *Figura 1.2*.

Figura 1.2 Modelo de Calidad del Producto de Software ISO/IEC 25010.2



Fuente: ISO/ICE 25010

Autor: ISO/ICE 25010

El presente trabajo de titulación tiene como enfoque el estudio de herramientas *Open Source* para pruebas de calidad sobre las características fiabilidad y rendimiento con sus respectivas subcaracterísticas. A continuación, se detallan de las características fiabilidad y rendimiento.

1.1.3 Fiabilidad del producto de Software

La fiabilidad es el grado en que el producto de software puede mantener un nivel de rendimiento especificado cuando se usa bajo condiciones especificadas. El desgaste o envejecimiento no ocurre en el software. Las limitaciones de fiabilidad en un producto de software se deben a fallos en los requisitos, diseño e implementación. Las fallas dependen de la forma en que se utilice el producto de software y de las opciones de programa seleccionadas. La característica fiabilidad tiene las siguientes subcacterísticas; madurez, disponibilidad, tolerancia a fallos, y recuperabilidad. A continuación, la descripción de cada subcaracterística:

- Madurez, indica el grado en que el sistema satisface las necesidades de fiabilidad en condiciones normales de funcionamiento.
- Disponibilidad, indica el grado en que un componente de software es operativo y disponible cuando se requiera para su uso.
- Tolerancia a fallos, indica el grado en que el producto de software puede mantener un nivel específico de actuación en casos de fallos de software o de la violación de su interfaz especificada.
- Recuperabilidad, indica el grado en que el producto de software puede volver a establecer un nivel específico de rendimiento y recuperar los datos directamente afectados en el caso de una falla.

1.1.4 Rendimiento del producto de Software

El rendimiento es el grado en que el producto de software proporciona un rendimiento adecuado, en relación con la cantidad de recursos utilizados, bajo condiciones establecidas. Estos recursos pueden estar relacionados o incluir a otros productos de software, al hardware del sistema y materiales como pueden ser dispositivos de almacenamiento externos, papel de impresión etc. La evaluación del rendimiento de un producto de software que es operado por un usuario se lo puede medir externamente por calidad de uso. El rendimiento tiene las siguientes subcaracterísticas:

- Comportamiento en el tiempo, indica el grado en que el producto de software proporciona los tiempos de respuesta y de procesamiento apropiado, y tasas de rendimiento al realizar su función, bajo las condiciones establecidas.
- Utilización de recursos, indica el grado en que el producto de software utiliza cantidades y tipos de recursos apropiados cuando el software lleva a cabo su función bajo condiciones establecidas.

1.1.5 Métricas de fiabilidad y rendimiento ISO/ICE 25023

Las métricas de calidad usadas son las que presentan la ISO/ICE 25023 [4], este estándar define métricas que permiten la medición cuantitativa de un producto software.

Las métricas de calidad externa se usan para medir la calidad del producto/sistema de software midiendo el comportamiento del sistema del cual forma parte, esta evaluación se la puede realizar durante las pruebas del ciclo de vida y/o en su etapa operativa. Es bueno tener en cuenta que los requisitos especificados por las medidas de calidad deben utilizarse como criterios cuando se evalúa un producto.

La clasificación de las métricas es en base a las características y subcaracterísticas de la norma ISO/ICE 25010 [3].

En este trabajo el uso de las métricas establecidas por la ISO/ICE 25023 cumplen dos roles importantes; primero permite la selección de las herramientas que cubren la mayor cantidad de métricas de fiabilidad y rendimiento, segundo la evaluación en caso de estudio.

1.1.5.1 Métricas de rendimiento

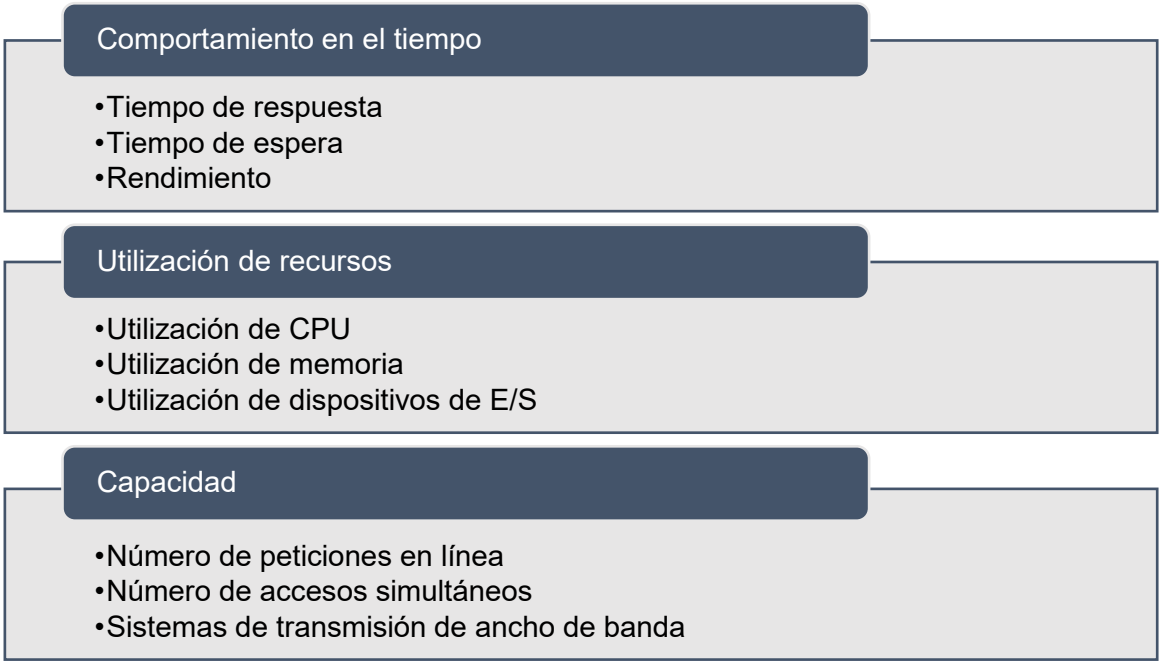
Las métricas de rendimiento deben tener la capacidad de medir el rendimiento en la relación con la cantidad de recursos utilizados en condiciones establecidas. Es importante aclarar que estos recursos pueden ser otros productos de software, el hardware del sistema y materiales externos como dispositivos de almacenamiento externos [2].

En la medición de rendimiento se debe tener condiciones establecidas, es decir se debe tener en cuenta la configuración del sistema y de hardware. La medición de rendimiento externo debe permitir medir atributos tales como el consumo de tiempo y el comportamiento de utilización de recursos del sistema informático incluyendo el software durante las operaciones o las pruebas. Es importante tener en cuenta el papel que desempeña factores

tales como CPU y memoria utilizada por otro software, tráfico de red y procesos de fondo programados.

Las métricas de rendimiento en la evaluación de calidad externa tienen un 13% de importancia. A continuación, en la *Figura 1.3* se puede observar un detalle de las subcaracterísticas de rendimiento con sus respectivas métricas.

Figura 1.3 Subcaracterísticas de rendimiento con sus respectivas métricas



Fuente: ISO/ICE 25023

Autor: Fernando Maila

Comportamiento en el tiempo

Las métricas de comportamiento en el tiempo deben permitir medir tales atributos como el comportamiento del tiempo del sistema informático incluyendo software durante la prueba o las operaciones. Las métricas de la subcaracterística comportamiento en el tiempo son: tiempo de respuesta, tiempo de espera y rendimiento.

Tiempo de respuesta

El tiempo de respuesta la duración de dar un comando para iniciar un lote de tareas hasta recibir la primera respuesta. El tiempo de respuesta incluye tiempo de procesamiento y tiempo de transmisión. En el caso del sistema de Internet u otro sistema de tiempo real, a

veces el tiempo de transmisión es mucho más largo. La fórmula para la medición de tiempo de espera es:

$$X = B - A$$

- A= Tiempo de envío de petición
- B = Tiempo en recibir la primera respuesta

Tiempo de espera

El tiempo de espera es el tiempo transcurrido en una computadora entre recibir un mensaje y enviar el resultado. A veces solo significa el tiempo usado para un programa de aplicación. La fórmula para la medición de tiempo de espera es:

$$X = B - A$$

- A= Tiempo cuando se inicia un trabajo
- B = Tiempo en completar el trabajo

Rendimiento

Es el tiempo que refleja cuántas tareas se pueden procesar por unidad de tiempo. La fórmula para la medición del rendimiento es:

$$X = A / T$$

- A = número de tareas completadas
- T = intervalo de tiempo

Utilización de recursos

Las métricas de utilización de recursos externos permiten medir tales atributos como el comportamiento de los recursos utilizados del sistema informático incluyendo el software durante las pruebas o la operación. Las métricas de la subcaracterística utilización de recursos son, utilización de CPU, utilización de memoria y utilización de dispositivos de E/S.

Utilización de CPU

La utilización de CPU permite medir cuánto tiempo de CPU se utiliza para realizar una tarea determinada. La fórmula para medir la utilización de CPU es:

$X = A$
• A = Cantidad de CPU que es usada para realizar una tarea

Utilización de memoria

La utilización de memoria permite medir cuánto espacio de memoria se utiliza para realizar una tarea determinada. La fórmula de la utilización de memoria es:

$X = A$
• A = Cantidad de memoria que es usada para realizar una tarea

Utilización de dispositivos de E/S

La utilización de dispositivo (s) de E/S permite medir cuánto tiempo consumen los dispositivos de E/S para realizar una tarea determinada. La fórmula para medir la utilización de dispositivo (s) de E/S es:

$X = A / B$
• A = tiempo de operación • B = la cantidad de dispositivo (s) de E / S ocupada para realizar una tarea

Capacidad

Las métricas de capacidad permiten medir el grado en que los límites máximos de un producto o parámetro del sistema cumplen los requisitos.

Número de peticiones en línea

Esta métrica permite medir cuántas solicitudes en línea pueden procesarse por unidad de tiempo. La fórmula es:

$X = A / T$
• A= Número máximo de peticiones online procesada • T = Tiempo de operación • T>0

Número de accesos simultáneos

Esta métrica permite medir cuántos usuarios pueden acceder al sistema simultáneamente en un momento determinado. La fórmula es:

$X = A / T$
• A = Número máximo de accesos simultáneos • T = Tiempo de operación • T > 0

Sistemas de transmisión de ancho de banda

Esta métrica permite medir cuánto es el valor límite absoluto de transmisión requerido para cumplir con las funciones. La fórmula es:

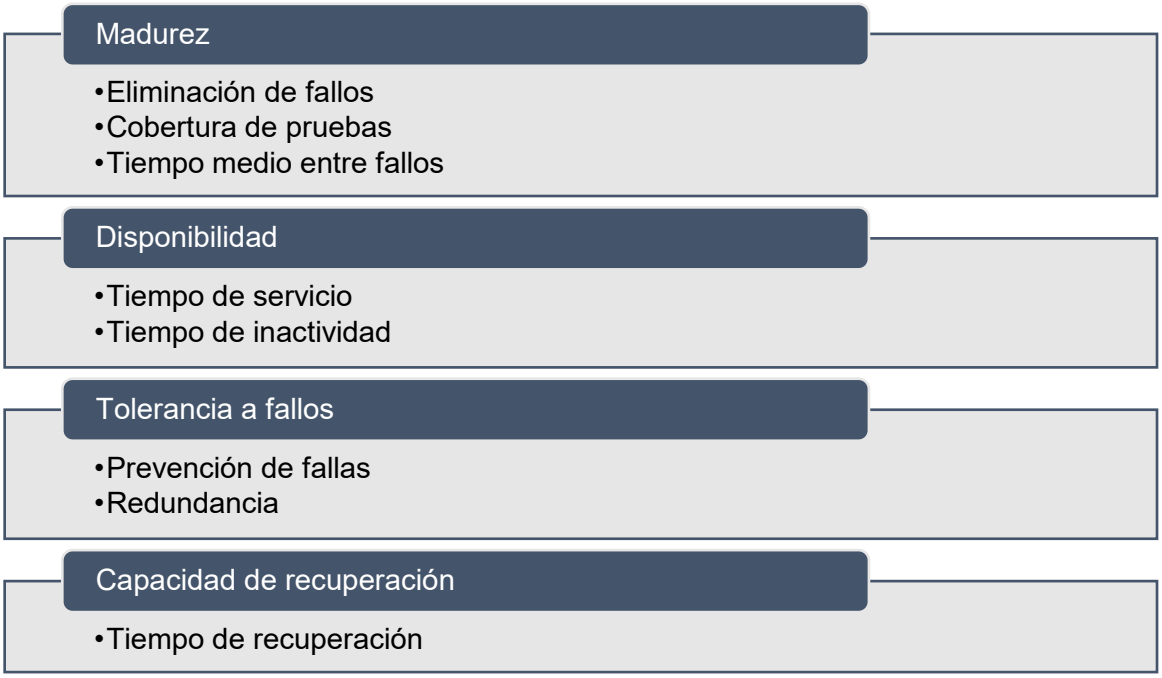
$X = A / T$
• A = Cantidad máxima de transmisión de datos • T = Tiempo de operación • T> 0

1.1.5.2 Métricas de fiabilidad

Las métricas de fiabilidad permiten medir los atributos relacionados con los comportamientos del sistema de la aplicación web es parte durante las pruebas de ejecución para indicar el grado de confiabilidad de la aplicación en ese sistema durante la operación. [2]

Las métricas de fiabilidad en la evaluación de calidad externa tienen un 15% de importancia. A continuación, en la *Figura 1.4* el detalle de las métricas de cada subcaracterística de fiabilidad.

Figura 1.4 Subcaracterísticas de fiabilidad con sus respectivas métricas



Fuete: ISO/ICE 25023

Autor: Fernando Maila

Madurez

Las métricas de madurez externa permiten medir atributos tales como la libertad de software de los fallos causados por fallas existentes en el propio software. Las métricas de esta subcaracterística son: eliminación de errores, cobertura de pruebas y el tiempo medio entre fallos.

Eliminación de fallos

La métrica de eliminación de fallos permite determinar qué proporción de fallos detectados se han corregido. La fórmula:

$$X = A / B$$

- A = Número de fallos corregidos en la fase de diseño / codificación / prueba.
- B = Número de fallos detectados en la revisión o prueba.

Cobertura de pruebas

La métrica cobertura de pruebas permite determinar cuánto de los casos de prueba requeridos se han ejecutado durante las pruebas. La fórmula es:

$X = A / B$
• A = Número de casos de prueba efectivamente realizados que representan el escenario de operación durante la prueba • B = Número de casos de prueba que deben realizarse para cubrir los requisitos

Tiempo medio entre fallos (MTBF)

La métrica de tiempo de tiempo entre fallos (*Mean Time Between Failures*) permite medir con qué frecuencia falla el funcionamiento del sistema o software. La fórmula es:

$X = A / B$
• A = tiempo de operación • B = número total de fallos realmente detectados.

Disponibilidad

La disponibilidad en la evaluación de calidad externa puede ser evaluada por la proporción del tiempo total durante el cual una aplicación está en estado activo. Por lo tanto, la disponibilidad es una combinación de madurez, tolerancia a fallos y capacidad de recuperación la cual se encarga de regular la duración del tiempo de inactividad después de cada falla. Las métricas de esta subcaracterística son el tiempo de servicio y el tiempo de inactividad.

Tiempo de servicio

La métrica de tiempo de servicio permite medir la proporción del tiempo de servicio del sistema se proporciona realmente. La fórmula para medir el tiempo de servicio es:

$X = A / B$
• A = tiempo de servicio del sistema realmente proporcionado • B = tiempo de servicio del sistema regulado en el calendario operativo

Tiempo de inactividad

La métrica de tiempo de inactividad permite medir cual es el tiempo promedio en que el sistema permanece indisponible cuando se produce un fallo. La fórmula para medir el tiempo de inactividad es:

$$X = A / B$$

- A = tiempo de inactividad total
- B = Número de averías observadas

Tolerancia a fallos

La subcaracterística tolerancia a fallos externa debe estar relacionada con la capacidad de la aplicación de mantener un nivel de rendimiento especificado en casos de fallos de operación o infracción de su interfaz especificada. Las métricas de esta subcaracterística son: prevención de fallas y redundancia.

Prevención de fallas

Esta métrica permite medir cuántos patrones de fallas fueron puestos bajo control para evitar fallas críticas y graves. La fórmula para medir la prevención de fallas es:

$$X = A / B$$

- A = Número de ocurrencia de fallas evitadas contra los casos de pruebas de fallas iniciales
- B = Número de casos de pruebas de fallas iniciales ejecutados durante las pruebas
- Dónde: B > 0

Redundancia

Esta métrica permite medir cuántos tipos de componentes del sistema se instalan de forma redundante para evitar fallos del sistema. La fórmula para medir la redundancia es:

$X = A / B$
• A = Número de componentes / sistemas instalados de forma redundante • B = Número total de componentes / sistemas instalados • Dónde B> 0

Capacidad de recuperación

La subcaracterística capacidad de recuperación es una medida que permite medir atributos tales como el software con el sistema capaz de restablecer su nivel adecuado de rendimiento y recuperar los datos directamente afectados en el caso de un fallo. Esta subcaracterística tiene la métrica tiempo medio de recuperación.

Tiempo medio de recuperación

Esta métrica permite medir el tiempo promedio que tarda el sistema en completar la recuperación del fallo. La fórmula para medir el tiempo medio de recuperación es:

$X = A / T$
• A = Número de casos en los cuales se ha observado que el sistema entró en recuperación. • T = Tiempo que le tomó al sistema en recuperarse • Dónde T > 0

1.1.6 Proceso de pruebas

La ISTQB (*International Software Testing Qualifications Board*) es la organización internacional que tiene como misión “definir y mantener un Cuerpo de Conocimientos que permita certificar a los probadores basándose en las mejores prácticas, conectando la comunidad internacional de pruebas de software y fomentando la investigación” [5].

Un proceso de pruebas permite la medición de la calidad de un producto de software en términos de los defectos encontrados. SI el producto de software pasa una prueba de calidad diseñada correctamente reduce el nivel general del riesgo de un sistema [6].

Un proceso de pruebas puede tener los siguientes objetivos:

- Identificar fallas en el producto de software.
- Aumentar la confianza en el nivel de calidad.
- Facilitar información para la toma de decisiones.
- Evitar la aparición de fallas.

Si bien la ISTQB recomienda que el proceso de pruebas debería estar implicado desde el inicio del ciclo de vida del software para evitar la introducción de defectos en el código, para este proyecto se parte ya con productos Open Source terminados.

Los siete principios de las pruebas de software

ISTQB ha propuesto una serie de principios que establecen ciertas pautas generales para las pruebas de software.

1. Las pruebas de productos de software pueden demostrar la existencia de defectos, pero no al contrario, es decir no pueden demostrar que el software no tenga defectos. Al realizar las pruebas de software se reduce la probabilidad de que existas defectos ocultos.
2. Realizar las pruebas de software con todas las combinaciones de entrada y salida es imposible, por lo que es más importante realizar un análisis de pruebas y priorizarlas.
3. Las pruebas tempranas se usan para la identificación de defectos en una etapa anticipada y se la debe realizar lo más pronto posible en ciclo de vida del software.
4. La agrupación de defectos por lo general se concentra en un número reducido de módulos, estos defectos deben ser detectados en una fase de pruebas previa al lanzamiento del software.

5. La paradoja de pesticida hace referencia a que, si se realizan repetitivamente las mismas pruebas una y otra vez, en consecuencia, se dejaran de encontrar defectos nuevos.
6. Las pruebas dependen del contexto en que se las realice, es decir, no es lo mismo probar un sistema financiero a un sistema de publicidad.
7. A pesar que se realicen todas las pruebas y se hagan todas las correcciones de los defectos detectados, no servirá de nada si no cumple las expectativas y necesidades de los usuarios.

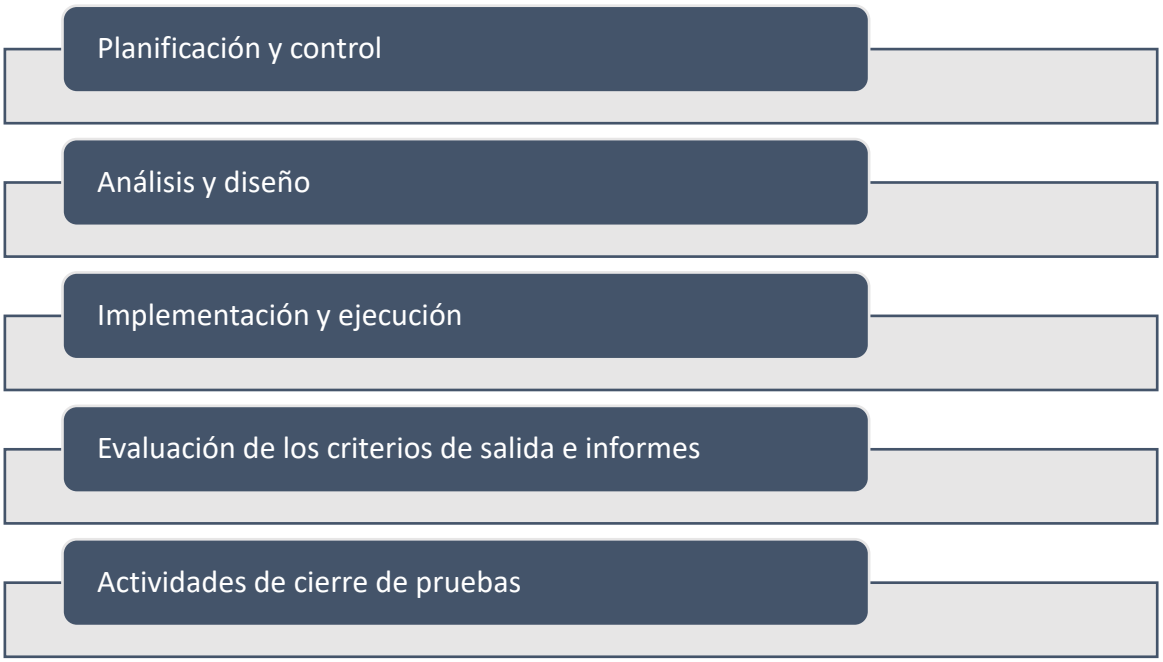
A pesar que en muchos de los puntos se referencia al ciclo de vida de un producto de software, cabe recalcar que en este proyecto las pruebas de calidad se las realiza a aplicaciones web finalizadas.

1.1.7 Proceso básico de pruebas

Un proceso de pruebas de software conlleva un grupo de actividades que no solo se enfoca en la ejecución de las pruebas, por lo tanto, hay que tomar en cuenta muchas aristas. El proceso de pruebas debe tener los tiempos necesarios que permitan planificar las pruebas, diseñar los casos de prueba, preparar la ejecución de las pruebas y finalmente evaluar los resultados.

En la *Figura 1.5* se puede observar un proceso de pruebas de software básico, este contiene las siguientes actividades básicas:

Figura 1.5 Proceso básico de pruebas



Fuente: ISTQB

Autor: Fernando Maila

Si bien se tiene una lista de actividades ordenadas secuencialmente, no significa que es obligatorio esperar a terminar una actividad para iniciar la siguiente. Dependiendo de las necesidades se las puede solapar o realizar 2 o más a la vez, adaptando las actividades a contexto del proyecto.

a) Planificación y control

En la etapa de planificación se definen los objetivos de las pruebas de software y la especificación de las actividades de pruebas con enfoque a cumplir los objetivos establecidos.

En el control se realiza un seguimiento en el cual se compara el progreso real con el plan establecido, además de informar el estado en el que se encuentran las pruebas. En esta actividad se deben incluir, en caso de haber existido, desviaciones con respecto a lo planificado.

b) Análisis y diseño de pruebas

En esta actividad los objetivos de las pruebas de software se transforman en condiciones de prueba y casos de prueba.

Esta actividad cuenta con una serie de tareas consideradas como principales, las cuales se señalan a continuación:

1. Revisar la base para las pruebas, esto se refiere a los requisitos, informes de análisis de riesgos, arquitectura, diseño y especificaciones de interfaz.
2. Evaluar la capacidad para ser probados los productos de software.
3. Identificar las condiciones de prueba y dar prioridades en base a un análisis.
4. Diseñar y priorizar los casos de prueba.
5. Identificar los datos de prueba necesarios que soporten las condicione de pruebas y los casos de pruebas.
6. Diseñar el entorno de pruebas e identificar herramientas e infraestructura en caso de ser necesario.

c) Implementación y ejecución de pruebas

En la actividad de implementación y ejecución de pruebas de software se especifican los procedimientos para las pruebas. Se inician con los casos de prueba en un orden determinado y se puede incluir información adicional de ser necesaria para la ejecución de las pruebas. Además, se configura el ambiente en el cual se ejecutan las pruebas.

La actividad de implementación y ejecución de pruebas tiene las siguientes tareas principales.

- Finalizar, implementar y dar prioridades a los casos de prueba.
- Desarrollar y dar prioridades a los procedimientos de pruebas.
- Revisar y verificar que el entorno de pruebas ha sido correctamente configurado.
- Ejecutar los procedimientos de pruebas de manera manual o mediante el uso de herramientas de ejecución de pruebas.
- Registrar los resultados obtenidos en la ejecución de las pruebas acompañado del detalle de la versión de software y las herramientas usadas en las pruebas.
- Realizar un reporte con las discrepancias encontradas en forma de incidencias y analizar en vistas de establecer que las causaron.

d) Evaluación de los criterios de salida e informe

La actividad previa a la finalización de todo el proceso es la evaluación de criterios de salida e informes. En esta actividad se evalúa la ejecución de las pruebas frente a los objetivos establecidos inicialmente.

Esta actividad consta con las siguientes tareas principales:

- Comprobar los registros de pruebas con los criterios de salida previstos en la planificación de las pruebas.
- Evaluar si se requiere que se realicen más pruebas o si se deberían modificar los criterios de salida previamente especificados.
- Realizar un resumen de las pruebas para las partes interesadas.

e) Actividades de cierre de pruebas

Esta actividad es el cierre del proceso de pruebas, en esta actividad se recopilan los datos obtenidos con el objetivo de consolidar la experiencia. Esta actividad es importante en el desarrollo de un proyecto de software, puede ser la actividad previa a un lanzamiento de un producto de software, la finalización de un proyecto de pruebas o la finalización de una versión de mantenimiento.

En la actividad de cierre de pruebas se incluyen las siguientes tareas principales:

- Comprobar cuales son los productos de software entregables previstos y cuales han sido definitivamente entregables.
- Finalizar los informes de incidencias o aportar modificaciones a aquellos que siguen abiertos.
- Documentar la aceptación del producto de software.
- Archivar los productos de soporte de prueba, el entorno de pruebas y la infraestructura de pruebas.
- Analizar las lecciones aprendidas en el proceso de pruebas para determinar futuras versiones y proyectos.
- Utilizar la información recopilada para mejorar la madurez de las pruebas.

1.1.8 Tipo de pruebas

En busca del cumplimiento de los objetivos es importante definir los tipos de pruebas que permiten medir métricas de rendimiento y fiabilidad, se ha considero las pruebas de carga y estrés.

Pruebas de Carga

Este tipo de pruebas permiten determinar y validar la respuesta de una aplicación cuando es sometida a una carga de usuarios, transacciones o procesos simultáneamente, tratando de emular un ambiente de producción real [8].

Pruebas de Estrés

En este tipo de pruebas se trata de cargar el sistema o los componentes del sistema hasta que llegan a los límites de funcionamiento. Esto permite encontrar el volumen de datos (o el tiempo) en que la aplicación deja de ser capaz de responder a las peticiones como se espera [8].

Dependiendo de las funcionalidades disponibles de cada herramienta se aplicará el tipo de prueba correspondiente. Cabe aclarar que las pruebas son de caja negra, las herramientas que realicen pruebas con el código de la aplicación serán descartadas.

2. METODOLOGÍA

En busca del cumplimiento de los objetivos planteados para este proyecto de titulación, se plantean las siguientes fases: determinación de un proceso de selección de herramientas, y descripción del caso de estudio.

La primera acción es realizar selección de las herramientas Open Source, que permite evaluar las métricas de las características de calidad fiabilidad y rendimiento. Para ello se realiza una búsqueda de las herramientas amplia en los diferentes repositorios, posteriormente se preseleccionan un grupo de herramientas baja ciertos criterios y finalmente se seleccionan las herramientas con los mejores perfiles y de ellas se elegirá una que se aplicará en el caso de estudio.

Como acción ejecutora, se utilizará la herramienta seleccionada para evaluar un caso de estudio. Para ello, se describe la aplicación web a evaluar, se presenta un plan de pruebas, su correspondiente diseño. Finalmente se aplicará la herramienta.

2.1. PROCESO DE SELECCIÓN DE HERRAMIENTAS

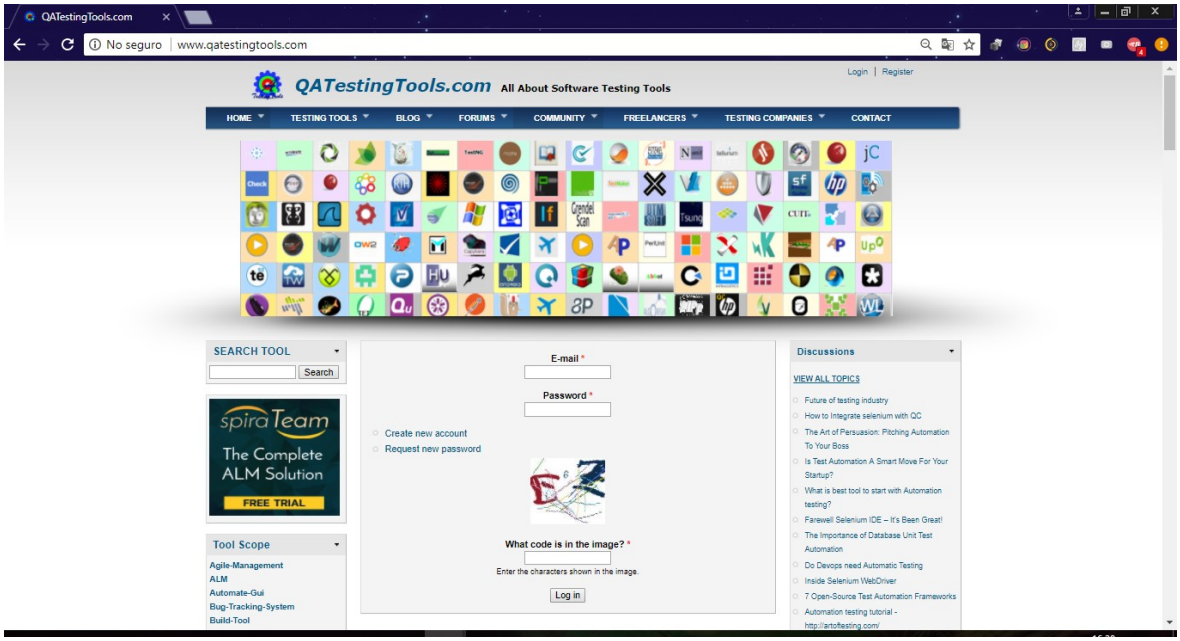
2.1.1 Búsqueda de herramientas Open Source para pruebas de fiabilidad y rendimiento.

La búsqueda de las herramientas es la actividad que inicia el proceso de selección de las herramientas, teniendo en cuenta que en esta actividad se agrupan todas las posibles herramientas a ser evaluadas. Al tener una lista de *n* herramientas se aplicarán ciertos criterios de selección con lo que se reducirá el universo, posteriormente se probarán las herramientas y finalmente se escogerá una que se aplicará al caso de estudio.

La búsqueda previa de las herramientas Open Source se la realiza en el buscador de Internet Google, posteriormente la búsqueda se centraliza en dos repositorios de productos de software Open Source que son *SourceForge* y *QA Testing Tools*. En el primer repositorio mencionado se encuentran aplicaciones para todas las necesidades de los usuarios, es decir, no es un repositorio específicamente de herramientas enfocadas a pruebas de calidad. Sin embargo, el segundo repositorio es una comunidad dedicado solo en productos para pruebas de software. Cabe recalcar que estos repositorios abarcan una gran cantidad de productos de software.

El repositorio *QA Testing Tools* (*Figura 2.1*) permite realizar búsquedas especificas mediante un menú con criterios como tipos de pruebas.

Figura 2.1 Repositorio *QA Testing Tools*



Fuente: *QA Testing Tools*

Autor: Fernando Maila

La búsqueda abierta de productos de software Open Source, tanto en Google como en los repositorios, que permitan realizar pruebas de fiabilidad y/o rendimiento tuvo como resultado una lista inicial que se presenta a continuación *Tabla 2.1*:

Tabla 2.1 Lista preliminar de herramientas Open Source para pruebas de fiabilidad y/o rendimiento.

No.	Herramienta
1	Canoo WebTest
2	CLIF
3	D-ITG
4	Fast Web Performance Test Tool
5	Funkload
6	Gatling
7	Grinder – Java Load Testing Framework
8	JMeter – Load and Performance tester
9	MStone
10	Multi-Mechanize – web performance and load testing framework
11	OpenSTA – Open Systems Testing Architecture
12	Performance Co-Pilot
13	Pylot – Performance & Scalability Testing of Web Services
14	SoapUI
15	s-watch
16	TestMaker
17	Tsung
18	Xceptance LoadTest

uno

Fuente: Internet

Autor: Fernando Maila

Es importante aclarar que para obtener esta lista de herramientas la búsqueda se la hizo en Internet tanto en los buscadores como en los repositorios. Con este número considerable de herramientas se procede a buscar información más detallada de cada herramienta que permitirá reducir el número de herramientas bajo ciertos criterios que posteriormente serán ampliados

2.1.2 Criterios de preselección de herramientas.

Con el fin de evaluar la herramienta Open Source con mejor perfil, se plantean ciertos criterios de selección. Si uno de estos criterios no es cumplido por la herramienta será descartada.

Para la evaluación de cada criterio se revisa la página oficial o el repositorio de la herramienta donde se obtendrá la información. Esta información se verá reflejada en una tabla (*Tabla 2.2*), en esta se marcará con un visto (✔) si cumple con el criterio y con una (**x**) si no lo cumple.

Los criterios que se considerarán para la preselección de herramientas son los siguientes:

- Ser Open Source.
- Fecha de la versión de la herramienta.
- La documentación.
- Un instalador disponible.
- Evalúe al menos una de las métricas de las características de fiabilidad y/o rendimiento con orientación a las aplicaciones web.

A continuación, se detalla cada uno de los criterios de preselección que se aplicarán a cada una de las herramientas.

Open Source

El propósito del proyecto es evaluar herramientas Open Source, aunque resulte evidente que las herramientas que serán evaluadas tienen que ser Open Source no está por demás aclararlo e investigarlo a profundidad.

Fecha de la última versión

La discriminación de aplicaciones por la fecha de última versión es debido a que las aplicaciones Open Source no garantizan su correcto funcionamiento, por lo que están en constante actualización de su software; Una herramienta que no tiene una versión reciente es porque ha sido abandonada por sus desarrolladores o la comunidad y en caso de presentarse algún inconveniente con la herramienta es posible que no se tenga una respuesta. Por lo antes mencionado se van a tomar en consideración herramientas que tengan como mínimo una última actualización del 1 de enero del año 2015 en adelante.

Documentación

La documentación es muy importante no solo en los productos de software Open Source, sino en todos los productos de software en general. Con el trascurrir de los años la práctica de documentación ha mejorado, esto ayuda a que los productos de software mejoren en sus fases de desarrollo y de mantenimiento [7]. Una documentación incompleta o deficiente es la causa de una gran cantidad errores y reduce la eficiencia en cada fase del desarrollo, pero en este caso tiene mayor relevancia para uso del producto de software [8].

En este proyecto de titulación primordial la existencia de la documentación de cada una de las herramientas, tanto para tener en claro el funcionamiento de las mismas como sus limitaciones. La herramienta que no tenga documentación o si la tienen no sea una buena documentación será descartada debido a que no es el objetivo del proyecto estar indagando que hacen sino poner a prueba su calidad.

Instalador disponible

Pareciera lógico que debe haber un instalador para probar la herramienta, pero existen casos en los repositorios en los que hay la descripción de las herramientas mas no existe el archivo de descarga, lo que la convierte en una herramienta inexistente para los usuarios. Por lo tanto, es importante verificar que están disponibles los instaladores de todas las aplicaciones de la lista preliminar.

La lista de herramientas encontradas en la búsqueda preliminar ya con los criterios antes mencionados se presenta a continuación en la *Tabla 2.2,* además se muestra un campo con el *link* de la herramienta.

Evalúa una métrica de fiabilidad y/o rendimiento

Las herramientas obtenidas en la búsqueda preliminar deben cumplir con el requisito de evaluar al menos una métrica ya sea de fiabilidad y/o rendimiento orientado a aplicaciones web. En caso de no cubrir al menos una métrica de las antes mencionadas, la herramienta será descarta.

Tabla 2.2 Herramientas y verificación de los criterios propuestos

No.	Herramienta	Open Source	Ultima Versión	Documentación	Instalador	Tipo de pruebas		Página oficial de la aplicación
						Fiabilidad	Rendimiento	
1	Canoo WebTest	✔	19/07/2016	✔	✔	x	x	http://webtest.canoo.com/webtest/manual/WebTestHome.html
2	CLIF	✔	6/4/2017	✔	✔	x	✔	http://clif.ow2.org/index.html
3	D-ITG	✔	03/06/2007	✔	✔	x	x	http://www.grid.unina.it/software/ITG/
4	Fast Web Performance Test Tool	✔	22/11/2016	✔	✔	x	✔	http://fwptt.sourceforge.net/
5	funkload	✔	10/3/20011	✔	✔	✔	✔	https://funkload.nuxeo.org/intro.html
6	Gatling	✔	20/04/2017	✔	✔	x	✔	http://gatling.io
7	Grinder – Java Load Testing Framework	✔	21/04/2015	✔	✔	x	✔	http://grinder.sourceforge.net/
8	JMeter – Load and Performance tester	✔	19/11/2016	✔	✔	x	✔	http://jmeter.apache.org/
9	MStone	✔	25/04/2013	x	✔	✔	x	https://www.qatestingtools.com/testing-tool/mstone
10	Multi-Mechanize – web performance and load testing framework	✔	01/01/2013	✔	✔	x	✔	https://multi-mechanize.readthedocs.io/en/latest/
11	OpenSTA – Open Systems Testing Architecture	✔	19/10/2007	✔	✔	x	✔	http://opensta.org
12	Performance Co-Pilot	✔	01/05/2017	✔	✔	x	x	http://pcp.io/index.html
13	Pylot – Performance & Scalability Testing of Web Services	✔	06/07/2009	x	✔	x	✔	https://code.google.com/archive/p/pylt/
14	SoapUI	✔	0/0/2014	✔	✔	x	✔	https://www.soapui.org/news/soapui-5-2-released.html
15	s-watch	✔	12/05/2014	x	x	✔	x	https://sourceforge.net/projects/s-watch/?source=directory
16	TestMaker	✔	15/10/2015	✔	✔	x	x	http://www.pushtotest.com/support.html
17	Tsung	✔	20/07/2015	✔	✔	✔	✔	http://tsung.erlang-projects.org/index.en.html
18	Xceptance LoadTest	✔	06/05/2017	✔	✔	x	✔	https://www.xceptance.com/en/

Fuente: Sitio web de cada aplicación

Autor: Fernando Maila

En la *Tabla 2.2* se representa la lista preliminar de herramientas Open Source para evaluar métricas de fiabilidad y/o rendimiento incluyendo los criterios de preselección planteados anteriormente. Se obtiene como resultado ocho herramientas que cumplen con los criterios de preselección, estas herramientas están resaltadas con color verde, las herramientas que no cumplen con los criterios de selección están marcadas de color naranja.

2.1.3. Selección de herramientas

La selección de herramientas que permitan evaluar las métricas de fiabilidad y herramientas se realiza valorando con diferente ponderación cada métrica. Tomando en cuenta que, hay herramientas que evalúan métricas de las dos características mencionadas, se plantea seleccionar obtenga la valoración más alta.

Para la selección final de las herramientas, se tomará como referencia la importancia que se le da a cada característica en la "MATRIZ DE CALIDAD PARA EVALUAR LA CALIDAD EXTERNA DE PRODUCTOS SOFTWARE EN EMPRESAS DE DESARROLLO DE SOFTWARE APLICANDO LA NORMA ISO/IEC 25000". Estos porcentajes de importancia se muestran a continuación en la *Tabla 2.3*.

Tabla 2.3 Porcentaje de importancia de las características rendimiento y fiabilidad

Característica	Porcentaje
Rendimiento	**13%**
Fiabilidad	**15%**

Fuente: ISO/ICE 25000
Autor: Fernando Maila

Con el objeto de conocer si cada herramienta cumple con una determinada métrica se hizo una revisión profunda de la documentación de cada herramienta. Esta información se muestra en la *Tabla 2.4* y se describe la herramienta con las métricas que cumple de forma resumida posteriormente.

Tabla 2.4 Herramientas seleccionadas con sus respectivas métricas

						No.	1	2	3	4	5	6	7	8
Características	% Importancia	%/100	Ponderación subcaracterística	Ponderación por métrica	Subcaracterística	Herramienta	CLIF	Fast Web Performance Test Tool	gatling	grinder	Jmeter	Performance Co-Pilot	tsung	Xceptance LoadTest
Rendimiento	13%	46.5%	1.86	0.62	Comportamiento del tiempo	Tiempo de respuesta	0.62		0.62		0.62			0.62
				0.62		Tiempo de espera	0.62		0.62		0.62			0.62
				0.62		Rendimiento					0.62			
			0.47	0.16	Utilización de recursos	Utilización de CPU	0.16						0.16	
				0.16		Utilización de la memoria	0.16						0.16	
				0.16		Utilización de los dispositivos de E/S								
			2.33	0.78	Capacidad	Numero de peticiones online (Max)	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78
				0.78		Número de accesos simultaneos (Max)	0.78	0.78	0.78	0.78	0.78	0.78	0.78	0.78
				0.78		Sistema de transmisión de ancho de banda			0.78		0.78			

Fiabilidad	15%	53.5%	1.18	0.39	Madurez	Eliminacion de Fallos								
				0.39		Cobertura de pruebas								
				0.39		Tiempo medio entre fallos								
			1.18	0.59	Disponibilidad	Tiempo de servicio							0.59	
				0.59		Tiempo medio de inactividad							0.59	
			1.18	0.59	Tolerancia a fallos	Prevención de fallas								
				0.59		Redundancia								
			1.18	1.18	Capacidad de recuperación	Tiempo medio de recuperación								
						Total	3.10	1.55	3.57	1.55	4.19	1.55	3.04	2.79

Fuente: Páginas oficiales de cada herramienta

Autor: Fernando Maila

La revisión literaria de cada herramienta es presentada en un pequeño fragmento con sus características más importantes. Las herramientas preseleccionadas son:

a) CLIF.
b) Fast Web Performance Test Tool.
c) Gatling.
d) Grinder – Java Load Testing Framework.
e) JMeter – Load and Performance tester.
f) Tsung.
g) Xceptance LoadTest.

CLIF

CLIF es una herramienta para pruebas de carga, esto incluye inyecciones de carga y es compatible con varios protocolos como son: HTTP, FTP, SIP. Además, permite medir el uso de recursos por parte de la aplicación como es: procesador, memoria, y red. Las características que más resalta de CLIF son las siguientes. [10]

- El usuario cuenta con 4 posibilidades de interfaces.
- Permite una monitorización gráfica y un control de los inyectores de carga.
- Plug-ins para a gestión de protocolos objetivos como son: HTTP, DNS, TCP, etc.
- Un motor de ejecución que permite gestionar millones de usuarios virtuales y un millón de peticiones por segundo.
- Inyectores de carga previstas UDP, TCP, FTP, HTTP (S), SIP, RTP, LDAP, JDBC, JMS, IMAP

En el sitio oficial de la herramienta se puede encontrar toda la documentación necesaria. Se puede encontrar ejemplos prácticos y manuales como son:

- Guía de instalación.
- Manual de inicio rápido.
- Manual de usuario.
- CLIF consola de Eclipse ayuda en línea.
- ISAC editor de escenarios ayuda en línea.
- ISAC manual de referencia plug-ins.
- Tutorial para CLIF y utilización ISAC.
- Usando CLIF desde servidor de automatización Jenkins.

Fast Web Performance Test Tool

Es una herramienta de evaluación de aplicaciones web, esta se enfoca en las pruebas de carga, además se puede grabar peticiones normales y Ajax.

Genera una clase C# que puede ser modificada, esta clase tiene funciones para peticiones HTTP que el usuario a grabado previamente. Puede grabar las acciones de navegación de diferentes exploradores lo que lo hacen versátil en este sentido.

En cuanto a su documentación, en la página oficial de la herramienta se puede ver un caso práctico de cómo grabar solicitudes y crear las definiciones de prueba. [11]

Gatling

La herramienta Gatling de se enfoca en las pruebas de rendimiento de aplicaciones web, estas pruebas consisten en cargar a la aplicación con muchos usuarios con comportamientos complejos.

Otra característica de la herramienta es que permite la revisión de los tiempos de respuesta a los requerimientos que se le hace a la aplicación. Los resultados obtenidos son presentados en informes donde se puede analizar los datos obtenidos con la ayuda de gráficos que permiten una mejor interpretación como se muestra en la Figura 2.2. [12]

Figura 2.2 Datos obtenidos con la herramienta Gatling

Fuente: Gatling

Autor: Gatling

La automatización de pruebas de Gatling es mediante scripts que permiten mantener escenarios y de prueba y a la vez facilita la automatización de las mismas. Gatling permite simular grandes cargas por segundo sobre la aplicación y obtener métricas con un alto margen de precisión.

Los desarrolladores de este proyecto, Gatling Corp, han acompañado a la herramienta con buena documentación la cual puede ser revisada en su página oficial (http://gatling.io), ahí se encuentra:

- Guía de Usuario
- Una guía de inicio rápido con la herramienta Gatling
- Un tutorial avanzado para el uso de la herramienta

Grinder – Java Load Testing Framework

La herramienta The Grinder es una aplicación Java, este framework permite la realizar pruebas de carga y de una manera fácil usar la carga distribuida utilizando muchas maquinas inyectoras. La licencia es Open Source BSD (*Berkeley Software Distribution*).
Las características que resaltan de la herramienta The Grinder las revisamos a continuación:

- Realiza pruebas de carga a las API Java, esto incluye los casos comunes de servidores web HTTP, SOAP, REST y servidores de aplicaciones CORBA, RMI, JMS, EJB así como los protocolos personalizados.
- A través de una consola gráfica se puede realizar múltiples inyecciones de carga que son monitoreadas y controladas, además ofrece edición de scripts centralizados y su distribución.
- Permite una gestión automática de conexiones de cliente y cookies, además de una aceleración de la conexión, un registro sofisticado y repetición interactiva entre navegador y el sitio web.

Los scripts utilizados para las pruebas son escritos en un lenguaje de programación dinámico, también especifica las pruebas a ejecutarse. Jython es el lenguaje de script predeterminado, este lenguaje es una implementación Java de Python.

El uso de estos scripts proporciona la siguiente ventaja: Se puede someter a una prueba cualquier código en lenguaje Java, Jython o cloiure. También se puede usar las bibliotecas Java disponibles con una gran variedad de sistemas y protocolos para realizar pruebas en Grinder.

JMeter – Load and Performance tester

Es una aplicación 100% java que permite medir el rendimiento de aplicaciones web. Las pruebas esta de rendimiento de recursos puede ser estáticas, dinámicas, o de aplicaciones web dinámicas. Entre sus principales características JMeter tiene las siguientes:

- Permite simular una carga pesada en el servidor, grupo de servidores, la red u objeto a probar, para analizar el rendimiento general con diferentes tipos de carga.
- Versatilidad de cargar y probar el rendimiento de muchos tipos diferentes de aplicaciones, servidores y protocolos.
- Tiene un IDE con todas las funciones de prueba, lo que permite un plan de prueba rápido de grabación, construcción y depuración.
- Se puede cargar las pruebas desde una línea de comandos, esto permite usar la herramienta desde cualquier sistema operativo compatible con java.
- Genera reporte HTML completo.

2.5.6 Performance Co-Pilot

Esta herramienta de código abierta tiene como principal función la supervisar y gestionar el rendimiento. La herramienta multiplataforma cuenta con colectores de información de rendimiento que puede extraer los datos de diferentes fuentes como el kernel o una base de datos. Cuenta con herramientas de supervisión que están en el colector o por separado. Una de las características más sobresalientes es que PCP (Performance Co-Pilot) puede estar ejecutándose remotamente con lo que se puede monitorear muchos hosts. La herramienta no viene sola, sino que son un conjunto de aplicaciones que forman el PCP base, a continuación, se describen cada una de ellas:

- Pmstat. - Produce un resumen ASCII de alto nivel del rendimiento del sistema.
- Pmie. - Un motor de inferencia que puede evaluar las reglas de acción de predicado para realizar alarmas y automatizar las tareas de administración del sistema.
- Pminfo. - Interrogación de métricas de rendimiento específicas y los metadatos que las describen.
- Pmlogger. - Genera archivos PCP de métricas de rendimiento adecuadas para reproducirlas por la mayoría de las herramientas de PCP.
- Pmval. - Informes periódicos simples para algunas o todas las instancias de una métrica de rendimiento, con control de tiempo de VCR opcional.
- Se pueden encontrar herramientas adicionales en el paquete de GUI de PCP estratificado.

- Pmchart. - Franja de gráficos para combinaciones arbitrarias de métricas de rendimiento.
- Pmdumptext. - Produce informes ASCII para combinaciones arbitrarias de métricas de rendimiento.

Tsung

Tsung es una herramienta Open Source bajo la licencia *GNU General Public v2* enfocada en pruebas de carga distribuida, esta herramienta es multiprotocolo y puede actuar sobre HTTP, WebDAV, SOAP, PostgreSQL, MySQL, LDAP and Jabber/XMPP. Una de sus principales funciones simular una gran cantidad de usuarios simultáneos desde una sola maquina con el fin de probar la escalabilidad y el rendimiento de la aplicación.

Tsung está desarrollado Erlang que es un lenguaje de programación orientado a la concurrencia, es ahí donde reside la gran ventaja de la herramienta. Tsung se basa en la Erlang OTP (*Open Transaction Plataform*) y hereda varias características de Erlang. A continuación, se revisa las características principales de Tsung:

- Simular una gran cantidad de usuarios en una maquina física. Además, que puede crear escenarios complejos con informes ampliados.
- La carga puede ser distribuida en los clientes.
- Supervisar el rendimiento de del sistema operativo (CPU, memoria y tráfico de red), esto mediante los agentes Erlang en servidores remotos o *SNMP.*
- Se puede simular varias sesiones con el fin de simular diferentes tipos de usuarios.

Xceptance LoadTest

La herramienta XLT permite ejecutar pruebas de regresión y carga para aplicaciones web que proporcionan acceso a través de HTTP / HTML. Cuenta con soporte JavaScript y las pruebas utilizan tecnologías web 2.0. Cuenta con un complemento para el navegador Mozilla Firefox, es una interfaz de usuario que permite scripting y el funcionamiento de casos de prueba. Los casos de prueba pueden ser exportados a Java y editados.

XLT soporta dos tipos de modelos de carga: un modelo de conteo de usuarios y un modelo de tasa de llegada.

El modelo de conteo de usuarios se caracteriza por ser un modelo de carga estático y no tiene una retroalimentación. Es por eso que este modelo puede ser adecuado para los

siguientes casos: una simple prueba de un usuario que sirve para evaluar el rendimiento básico, una prueba de rendimiento bajo una carga alta pero previsible, o una prueba que puede ser fácilmente repetible en la que la carga no pueda ser influenciada por el sistema que está bajo prueba.

EL modelo de tasa de llegada está basado en la retroalimentación, es decir, el número de usuarios simultáneos no es estático e influido por el tiempo de respuesta. Este modelo de carga es adecuado para probar si una aplicación web puede manejar un cierto número de transacciones en un intervalo de tiempo determinado.

Con base a la Tabla 2.4, en la siguientes sección se describe el uso y pruebas realizadas con las 3 herramientas de mayor puntuación.

2.1.4. Instalación y uso de herramientas

El proceso de prueba de cada herramienta inicia con la descarga de los instaladores o ejecutables de las páginas oficiales.

Tabla 2.5 Herramientas con su respectivo link de descarga.

Herramienta	Link de descarga
JMeter	http://jmeter.apache.org/download_jmeter.cgi
CLIF	https://forge.ow2.org/project/showfiles.php?group_id=57
Gattling	https://gatling.io/

Fuente: Página web de cada herramienta

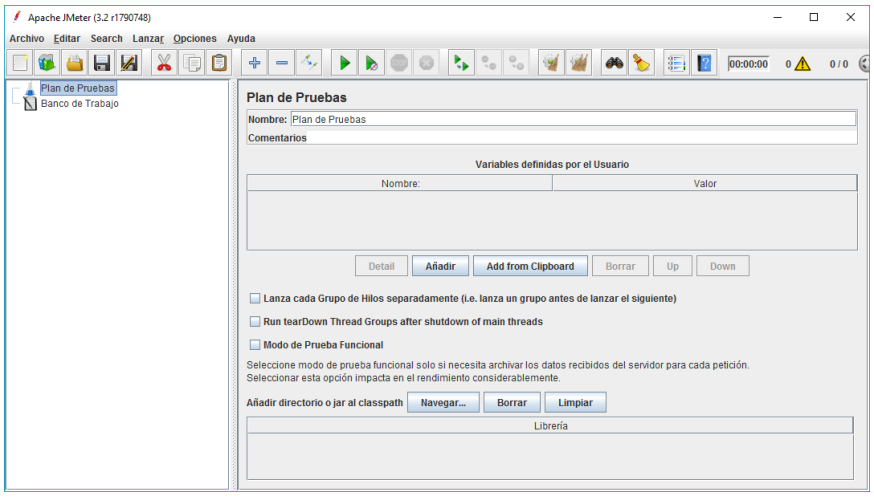
Autor: Fernando Maila

2.1.4.1 JMeter

Se descarga un archivo comprimido que contiene la aplicación, se procede a descomprimir el archivo, el ejecutable se encuentra en la ruta "..\apache-jmeter-3.2\bin\ApacheJMeter.jar".

Al ejecutar la aplicación se muestra una interfaz como en la de la Figura 2.3.

Figura 2.3 Interfaz JMeter



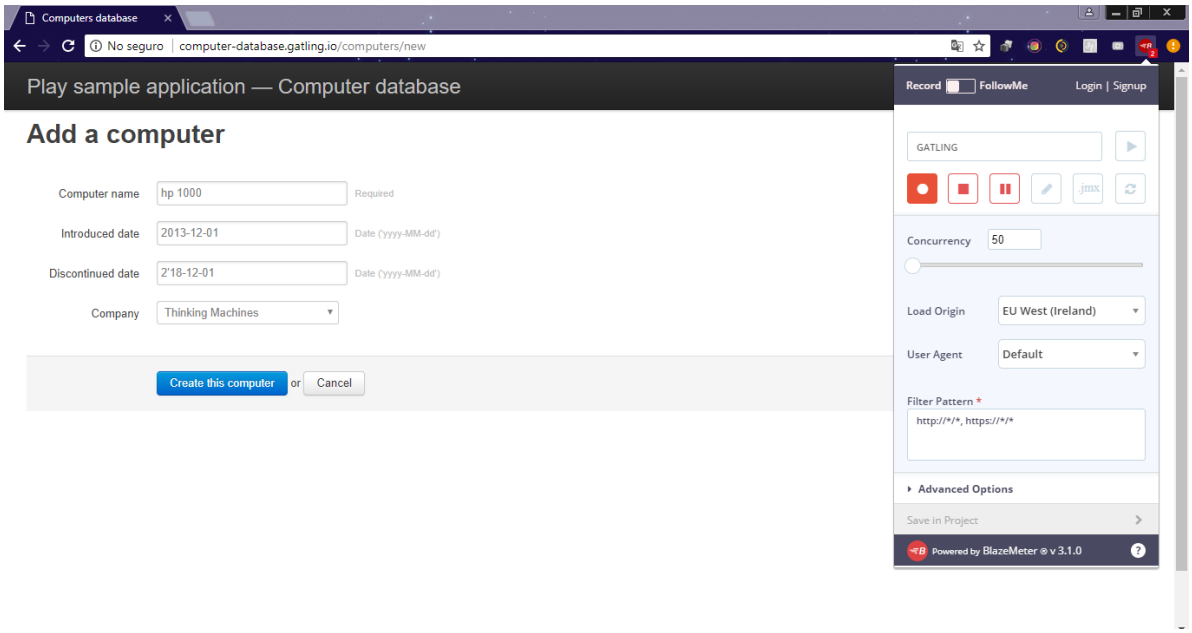
Fuente: Fernando Maila

Autor: Fernando Maila

Para la prueba se ha seleccionado una aplicación web para pruebas de gatling, esta permite hacer búsquedas e ingresas productos. El link de la aplicación es http://computer-database.gatling.io/computers .

Se procede a la grabación del script en el cual se realiza un ingreso de los datos de un computador, esto se lo realiza con la ayuda de BlazeMeter como se muestra en la Figura 2.4, un complemento de JMeter.

Figura 2.4 BlazeMeter, complemento de JMeter

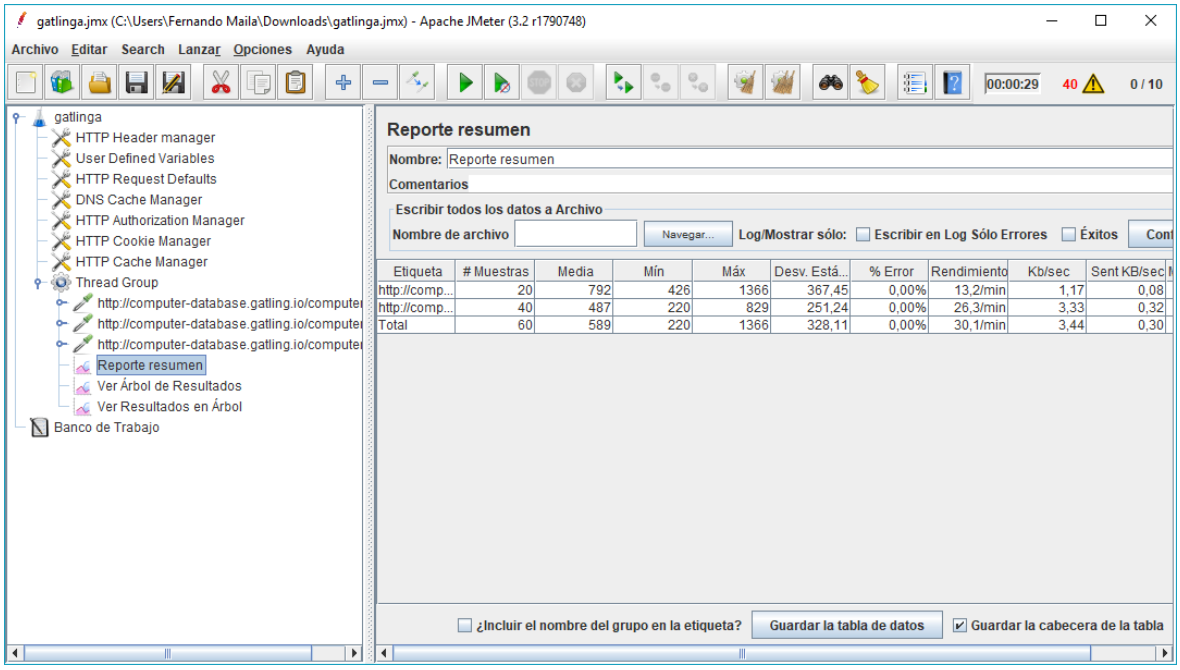


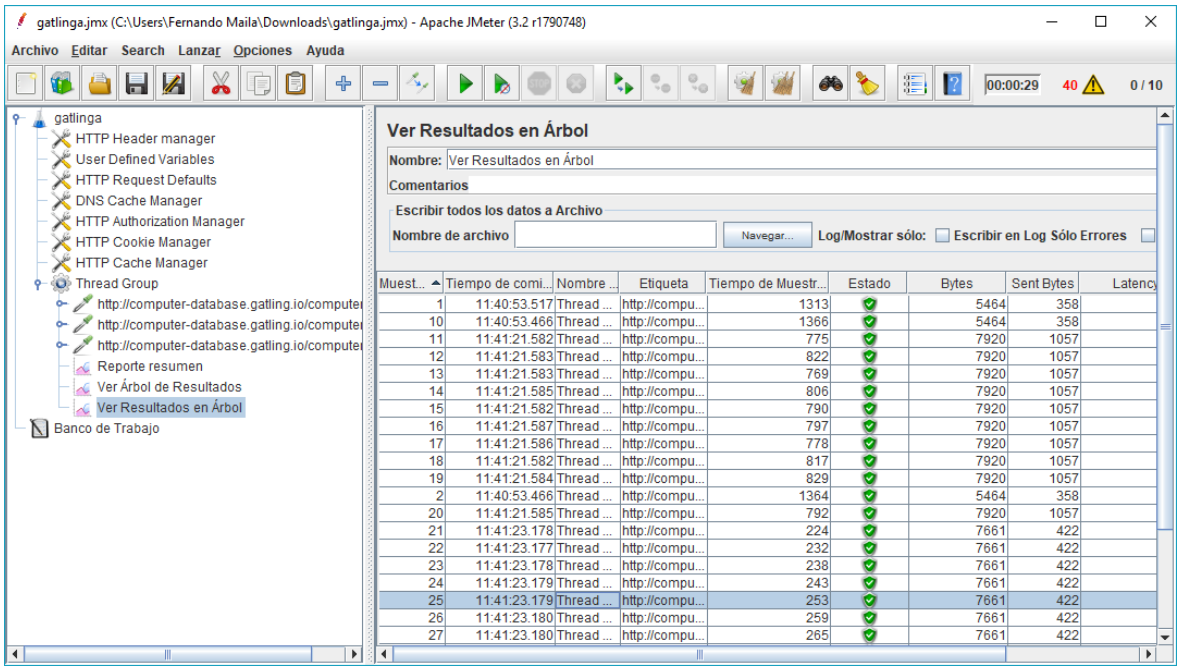
Fuente: Fernando Maila

Autor: Fernando Maila

Se carga el script en JMeter y se ejecuta con una carga de 100, los resultados se presentan en la Figura 2.5.

Figura 2.5 Uso de JMeter con carga de 10 peticiones





Fuente: Fernando Maila

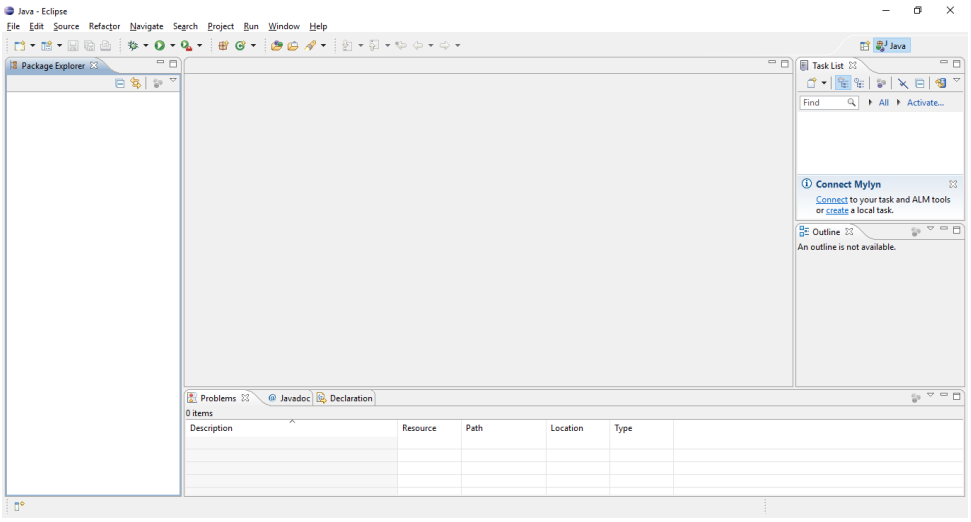
Autor: Fernando Maila

2.1.4.2 CLIF

Se descarga un archivo comprimido que contiene la aplicación, se descarga la versión Clif 2.3.5, se procede a descomprimir el archivo, el ejecutable se encuentra en la ruta "..\clif-

2.3.5-eclipseconsole\clif-console.exe". Al ejecutar la aplicación se muestra una interfaz de Eclipse como en la de la Figura 2.6.

Figura 2.6 Interfaz CLIF

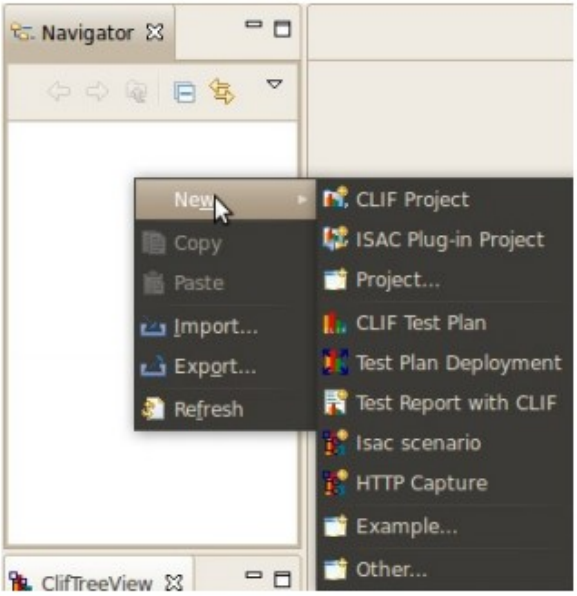


Fuente: Fernando Maila

Autor: Fernando Maila

Se presenta un problema a ejecutar la herramienta, al crear un nuevo proyecto debe mostrarse una interfaz como en la Figura 2.7.

Figura 2.7 Manual creación de proyecto con CLIF

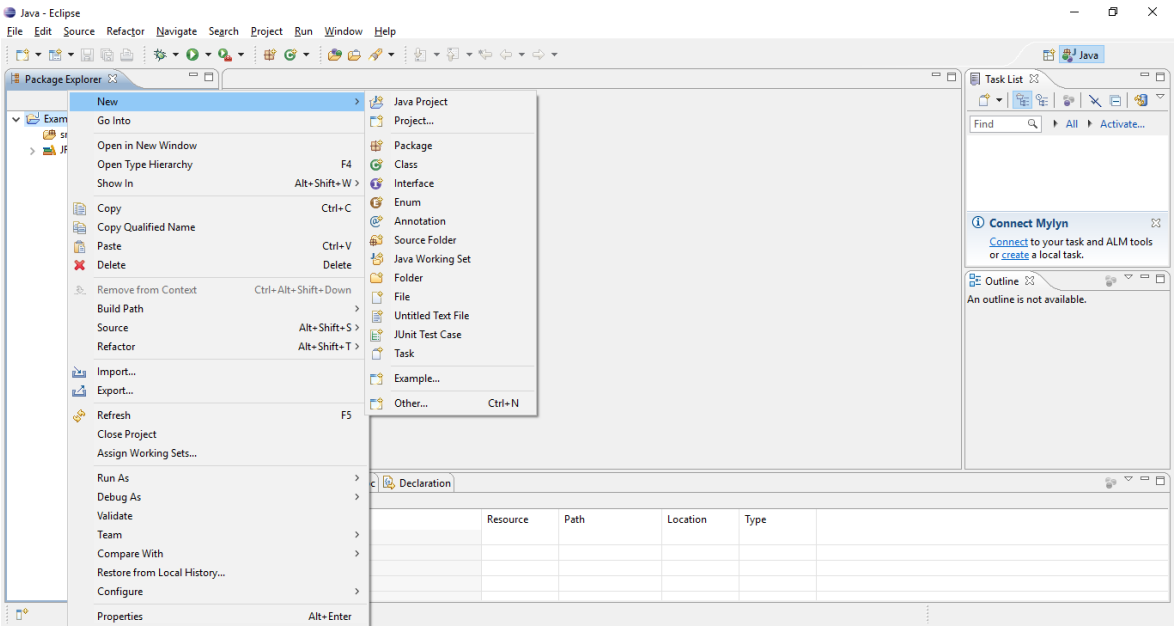


Fuente: Manual CLIF

Autor: Fernando Maila

En la interfaz de la aplicación descargada no se muestra el menú con las opciones de CLIF, se muestra como un Eclipse de desarrollo normal Figura 2.8.

Figura 2.8 Creación de proyecto con CLIF



Fuente: Fernando Maila

Autor: Fernando Maila

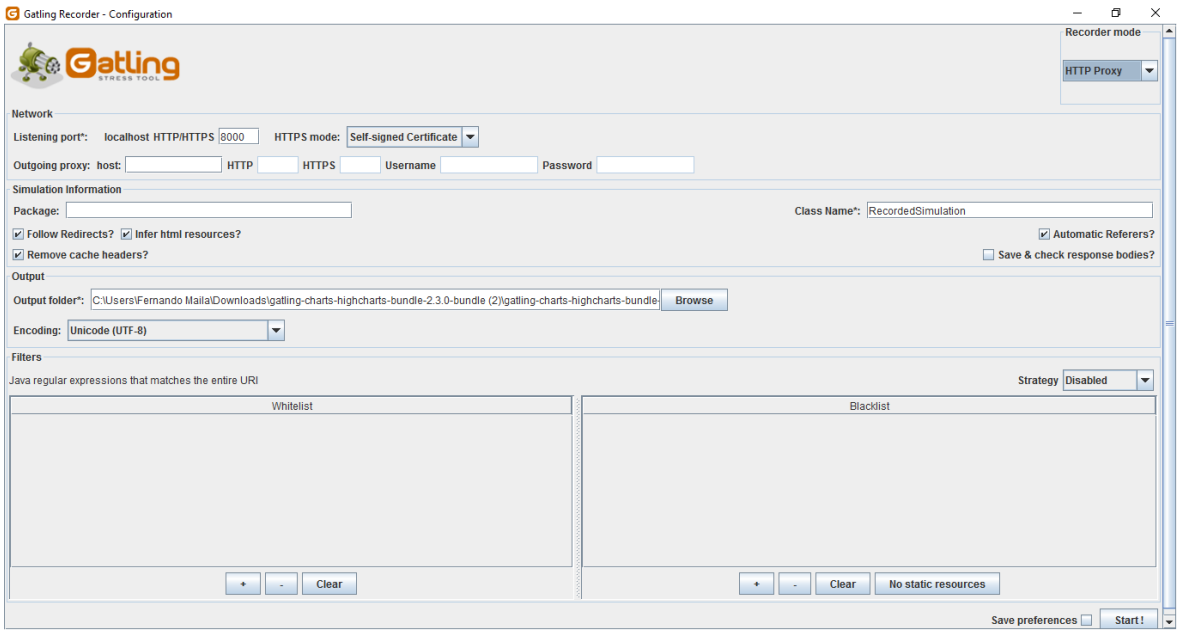
Una de las soluciones es cargar manualmente los plugins, para lo cual se ingresa de nuevo en el repositorio de de CLIF. Una vez descargados desde la aplicación Eclipse, ingresar en el menú *Help*, luego en Install New Software. A intentar instalar se produce un error. Esta aplicación se intentó instalar siguiendo el manual en formato:
Video: https://www.youtube.com/watch?v=wfTGi1W97qk
PDF: http://clif.ow2.org/doc/OW2Con10-tutorial-CLIF.pdf
Debido a los problemas presentados, se descarta esta aplicación.

2.1.4.3 Gattling

Se descarga un archivo comprimido que contiene la aplicación, se procede a descomprimir el archivo, el ejecutable se encuentra en la ruta "..\gatling-charts-highcharts-bundle-2.3.0\bin\recorder.sh".

Al ejecutar la aplicación se muestra una interfaz como en la de la Figura 2.9.

Figura 2.9 Interfaz Gatling

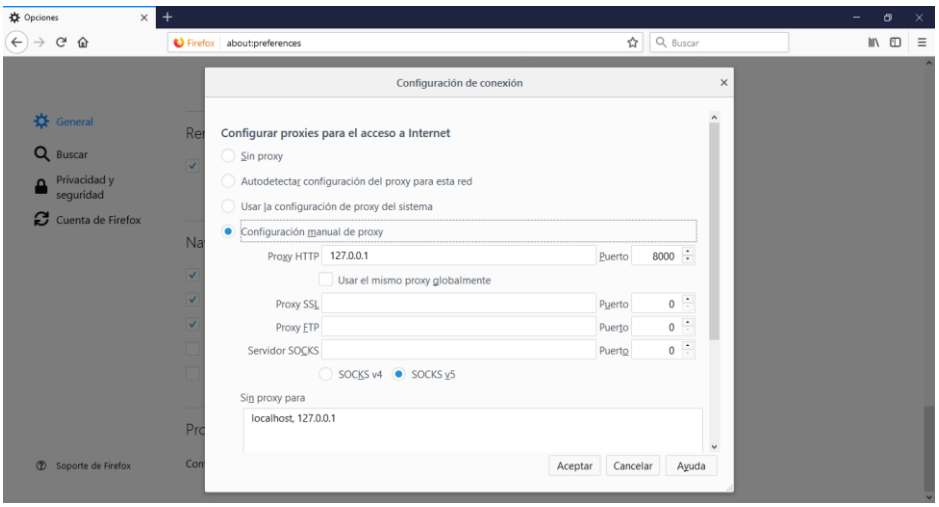


Fuente: Fernando Maila

Autor: Fernando Maila

La aplicación por defecto escucha en el puerto 8000. Para grabar la prueba se debe configurar el navegador. Se crea un proxy manual en la configuración del navegador como se muestra en la Figura 2.10.

Figura 2.10 Configuración de proxy



Fuente: Fernando Maila

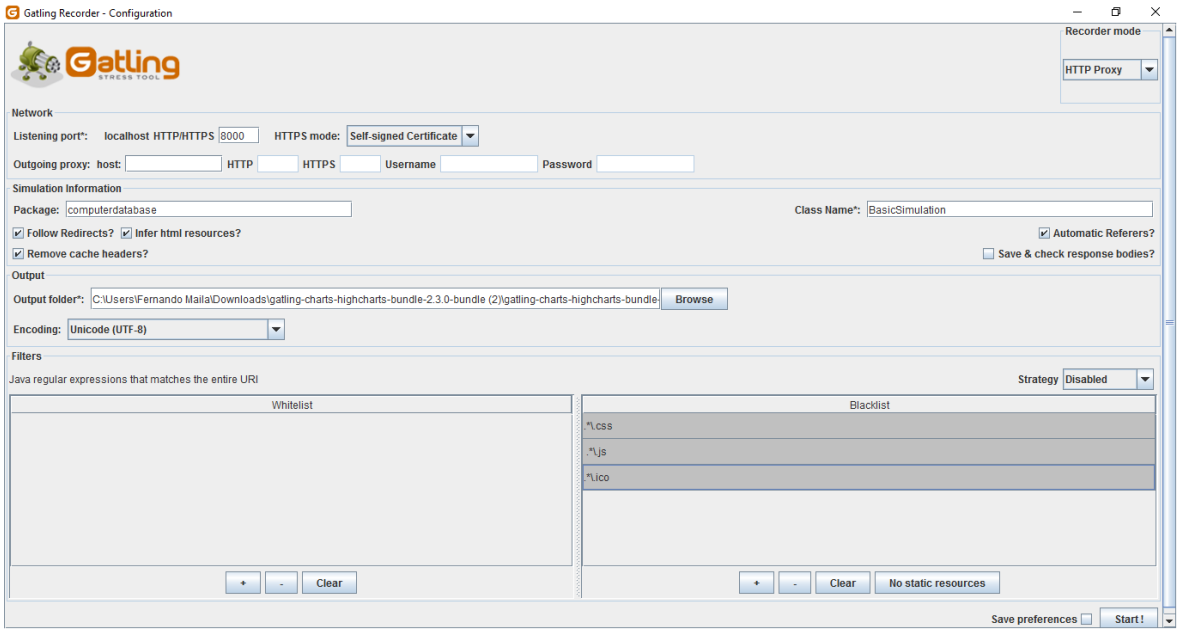
Autor: Fernando Maila

Previo a grabar el script se ingresa los parámetros de configuración:

- Nombre del paquete computerdatabase
- Nombre del test BasicSimulation
- Marcar Follow Redirects?
- Marcar Automatic Referers?
- Se selecciona como estrategia de filtrado Black list first
- Se usan los siguientes filtros en la black list “.*\.css”, “.*\.js” and “.*\.ico” para excluir llamadas a estáticos.

La interfaz con los parámetros se la puede apreciar en la Figura 2.11.

Figura 2.11 Interfaz Recorder de Gatling

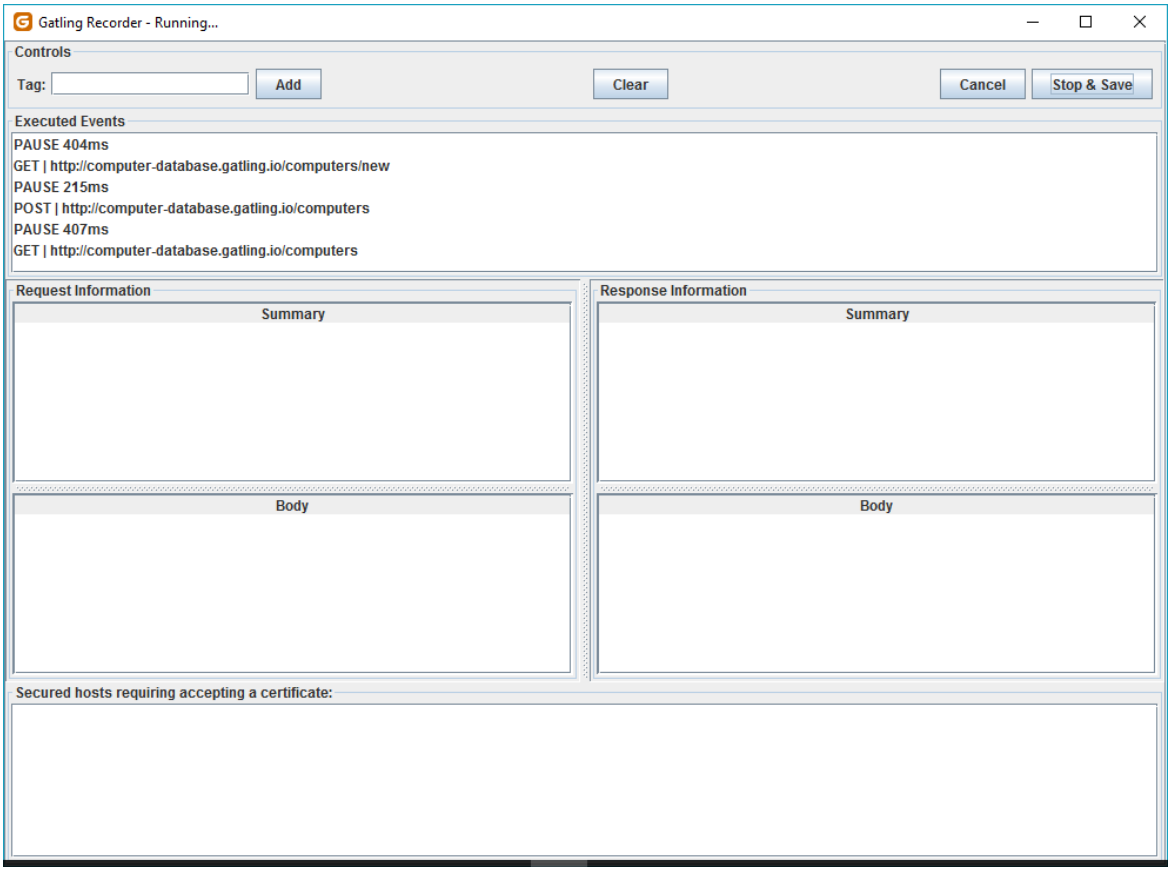


Fuente: Fernando Maila

Autor: Fernado Maila

Se procede a grabar el script ingresando datos de un computador en la página de pruebas de Gatling http://computer-database.gatling.io/computers. En la Figura 2.12 se puede observar cómo va registrando los eventos Gatling.

Figura 2.12 Recorder en funcionamiento



Fuente: Fernando Maila

Autor: Fernando Maila

Para realizar un ejercicio con la misma mis carga que la herramienta JMeter se debe modificar el archivo generado con el Recorder, para ello ingresar en el directorio "..\gatling\user-files\simulations", abrir el archivo "BasicSimulation.scala" en un editor de texto y modificar el numero de usuarios como se muestra en la Figura 2.13.

Figura 2.13 Modificación de carga de usuarios

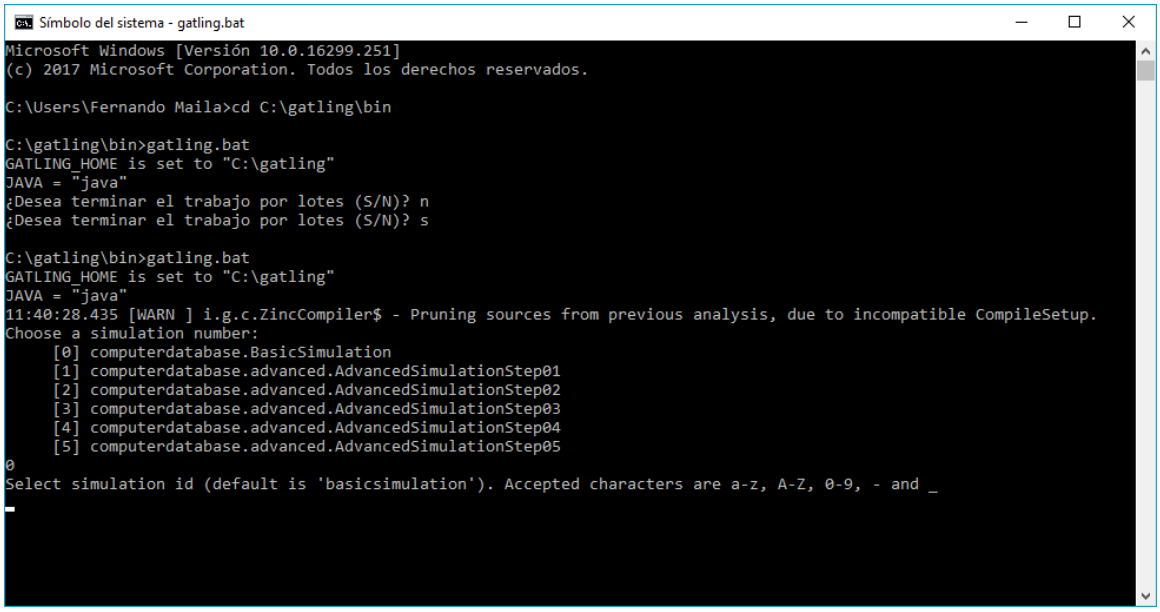
```
package computerdatabase

import scala.concurrent.duration._

import io.gatling.core.Predef._
import io.gatling.http.Predef._
import io.gatling.jdbc.Predef._

class BasicSimulation extends Simulation {

    val httpProtocol = http
        .baseURL("http://computer-database.gatling.io")
        .inferHtmlResources()
        .acceptHeader("text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8")
        .acceptEncodingHeader("gzip, deflate")
        .acceptLanguageHeader("es-MX,es;q=0.8,en-US;q=0.5,en;q=0.3")
        .userAgentHeader("Mozilla/5.0 (Windows NT 10.0; WOW64; rv:58.0) Gecko/20100101 Firefox/58.0")

    val headers_0 = Map("Upgrade-Insecure-Requests" -> "1")



    val scn = scenario("BasicSimulation")
        .exec(http("request_0")
            .get("/computers/new")
            .headers(headers_0))
        .pause(11)
        .exec(http("request_1")
            .post("/computers")
            .headers(headers_0)
            .formParam("name", "Frenos")
            .formParam("introduced", "1999-01-18")
            .formParam("discontinued", "2010-01-18")
            .formParam("company", "1"))

    setUp(scn.inject(atOnceUsers(10)).protocols(httpProtocol)
}
```

Fuente: Fernando Maila

Autor: Fernando Maila

Para ejecutar el script ingresar en el directorio de la herramienta y ejecutar “..\gatling-charts-highcharts-bundle-2.3.0\bin\gatling.sh”. Se muestra una interfaz con la de la Figura 2.14.

Figura 2.14 Ejecución de Gatling.sh

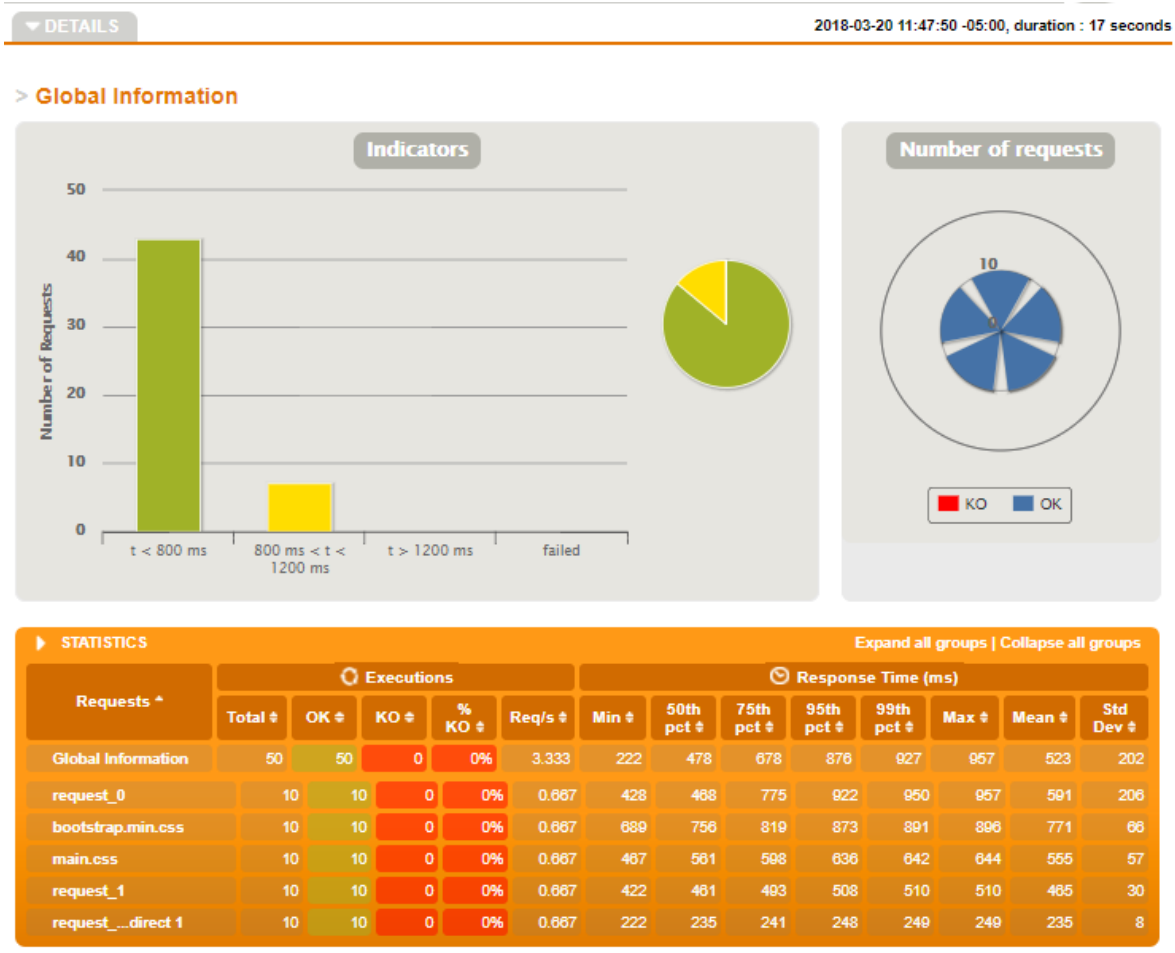


Fuente: Fernando Maila

Autor: Fernando Maila

Para observar los resultados de una manera grafica ingresar en el directorio “gatling\results*nombredelasimulación*” y ejecutar el archivo Index.

Figura 2.15 Resultados Gatling



Fuente: Fernando Maila

Autor: Fernando Maila

Gatling presenta los resultados gráficamente en muchas tablas diferentes, en la Figura 2.15 se puede observar las peticiones que se realizaron en los diferentes intervalos de tiempo.

Para realizar una comparación entre los resultados obtenidos por JMter y Gatlin se presenta la Tabla 2.6 en la que se pueden de manera resumida lo obtenido por las dos herramientas con una carga de 10 usuarios.

Tabla 2.6 Resumen de resultados de las herramientas ejecutadas

	Tiempo de respuesta (ms)	Tiempo de espera (s)	Rendimiento (peticiones/min)	Número de peteciones (peticiones/s)	Número de accesos simultaneos (num accesos/s)	Sistemas de transmisión de ancho de banda (Kb/s)
JMeter	1313	29	30,1	2,06	0,34	3,44
Gatling	597	17	-	3,33	0,52	-

Fuente: Fernando Maila

Autor: Fernando Maila

Los valores obtenidos son bastante cercanos por lo que se demuestra una consistencia en las herramientas.

Los valores reflejados en la tabla están en la unidad milésimas de segundo.

2.2 Caso de estudio

Para aplicar el proceso de pruebas de aplicaciones web, se utilizará una aplicación desarrollada para una PYME que se describe a continuación.

2.3.1 Descripción caso de estudio

La aplicación de la herramienta de prueba Open Source se lo hará en una aplicación web que ha sido elaborada para la mecánica “METALMOTORS S.A” ubicada en la ciudad de Quito. Esta empresa ofrece productos y servicios tanto en área automotriz como en el área industrial.

La empresa con el objetivo de estar en la vanguardia de la tecnología ha decidido implementar un sistema de que le permite llevar un control de sus productos, ventas, clientes, proveedores e inventario.

Se aplicará diferentes casos de prueba a la aplicación web con diferente carga con lo que se podrá analizar los datos obtenidos.

2.3.2. Aplicación proceso de pruebas

3.2.1 PLAN DE PRUEBAS

3.2.1.1 Objetivo

Implementar pruebas usando la herramienta Open Source, que fue seleccionada previamente, en la aplicación web de inventario y ventas de la empresa “IL METALMOTORS” para determinar si la herramienta si es una buena alternativa para las MICROPYMES de desarrollo de aplicaciones web.

3.2.1.2 Descripción del sistema a probar

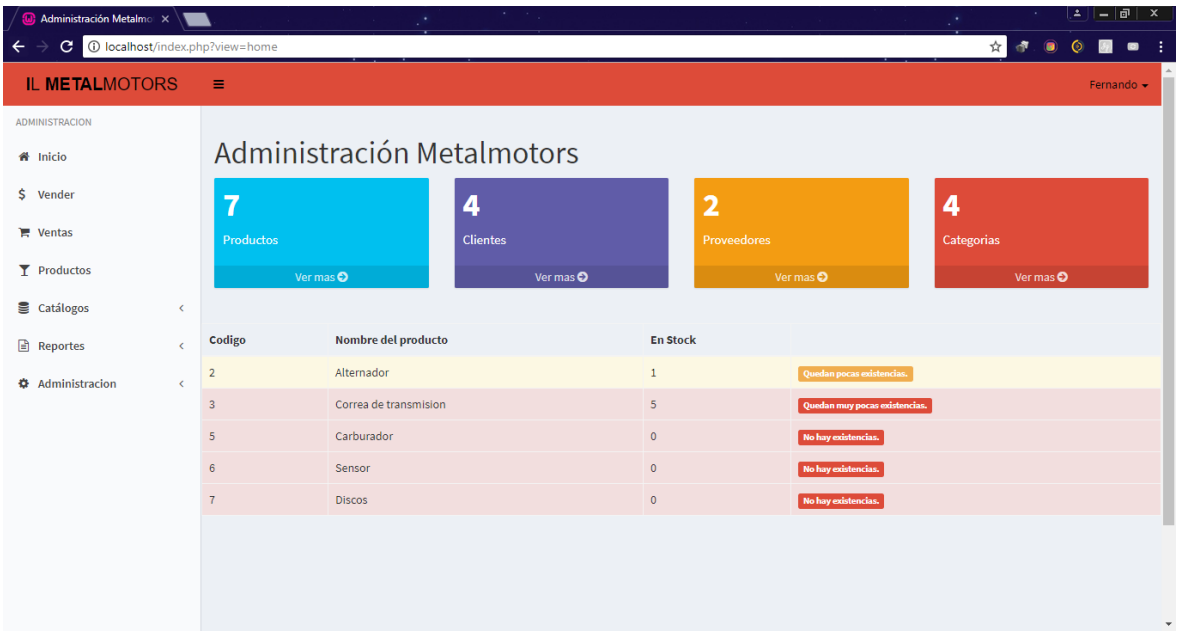
La empresa “IL METALMOTORS” se dedica a desarrollar, producir, comercializar y distribuir productos y servicios metalmecánicos y automotrices de excelente calidad con una ingeniería acorde a los requerimientos los clientes o el mercado.

La aplicación web de inventario de la empresa “IL METALMOTORS” permite llevar un control de: ventas, clientes, proveedores, inventario y la generación de reportes. Los módulos de la aplicación son:

- Autenticación
- Vender
- Ventas
- Productos
- Catálogos
- Reportes
- Administración

La aplicación está desarrollada en el lenguaje PHP y conectada al gestor de base de datos MySQL. En la *Figura 3.1* se puede observar la página principal de la aplicación con sus módulos.

Figura 3.1 Página principal de la aplicación "IL METALTOURS"



Fuente: Aplicación IL METALTOURS

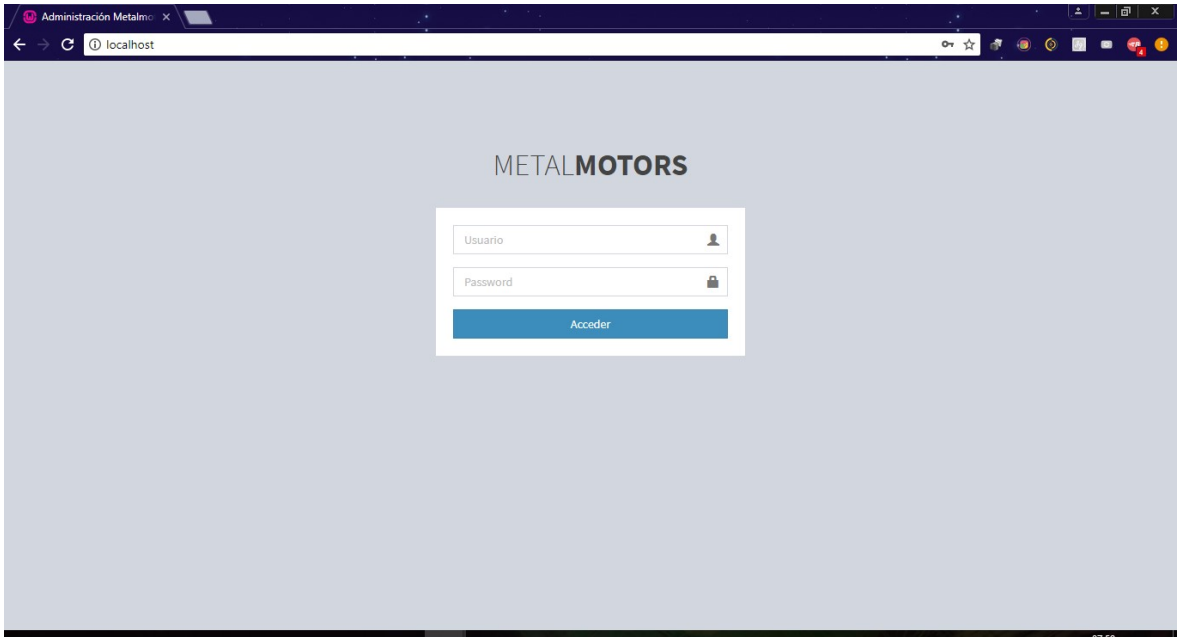
Autor: Fernando Maila

3.2.1.3 Módulos del sistema a probar

La aplicación "IL METAL MOTORS" cuenta con seis módulos para la administración del negocio. Los módulos que serán probados por las aplicaciones Open Source son: Autenticación, Productos y Reportes.

El módulo de autenticación es a seguridad de acceso a la aplicación web, si no se autentica correctamente no podrá acceder. La interfaz de este módulo se muestra en la *Figura 3.2*, este será el primer módulo que será sometido a las pruebas con la herramienta Open Source.

Figura 3.2 Módulo Autenticación

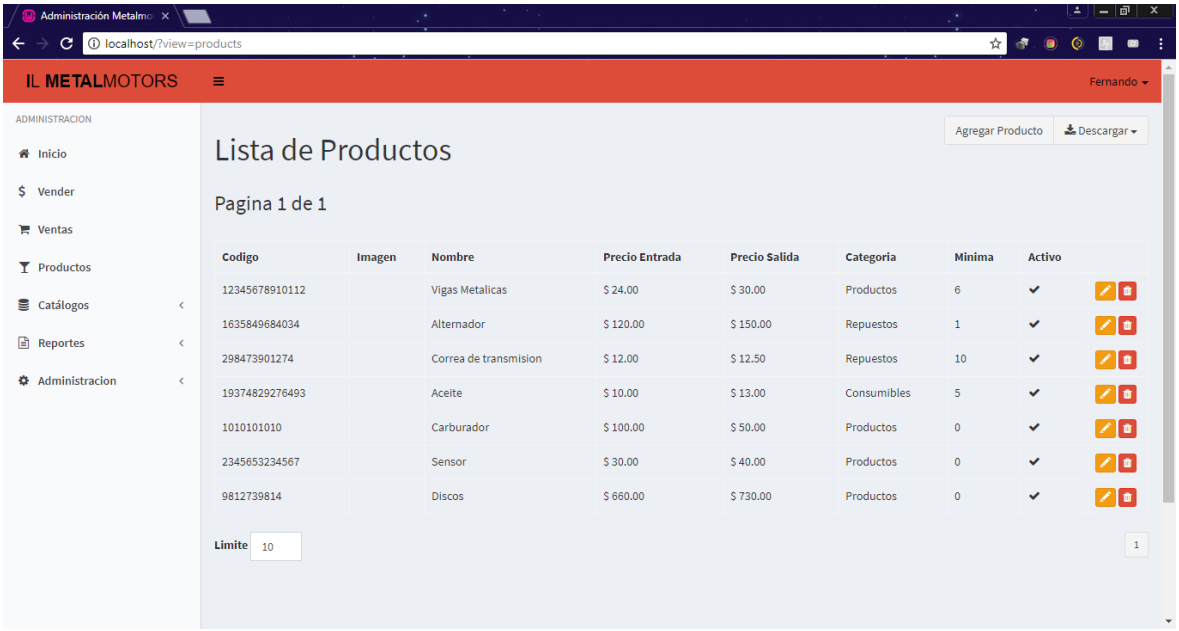


Fuente: Aplicación IL METALTOURS

Autor: Fernando Maila

El módulo *Productos* presenta una lista de productos agregados como se puede observar en la *Figura 3.3*, pero también tiene el Botón Agregar Productos el cual despliega un formulario con los campos de información de un producto como se puede observar en la *Figura 3.4*. El formulario de ingreso de producto será sometido a pruebas.

Figura 3.3 Módulo Productos

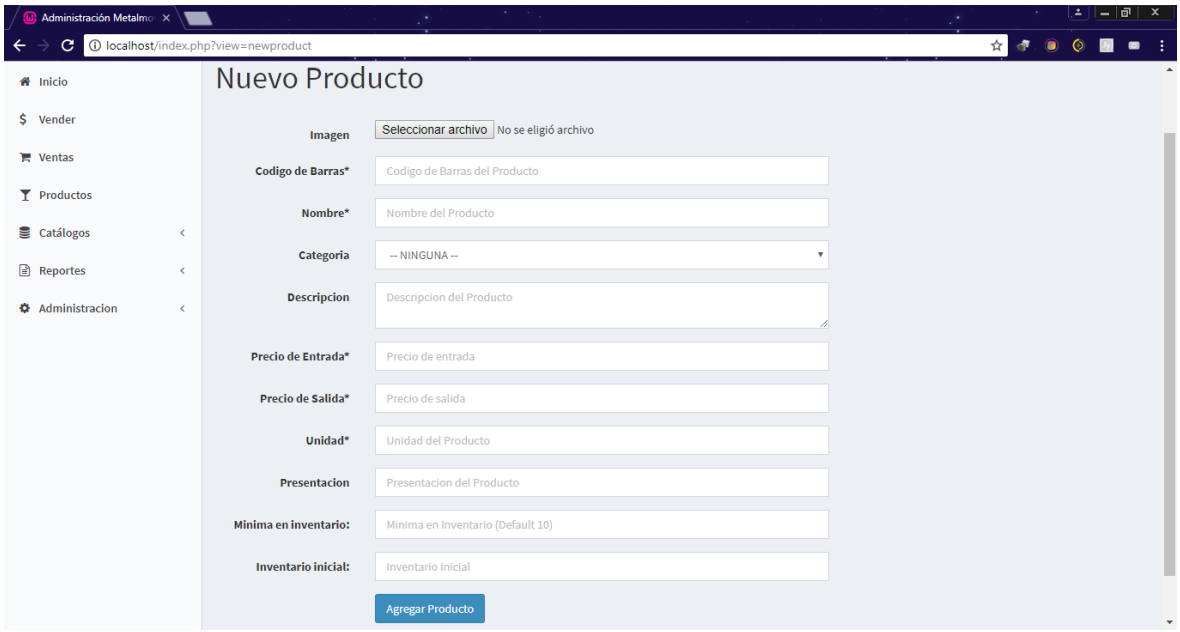


Fuente: Aplicación IL METALTOURS

Autor: Fernando Maila

El formulario para ingresar productos se muestra en la *Figura 3.4*, tiene campos que son obligatorios de llenar como el código de barras, nombre, precio de entrada, precio de salida y unidad.

Figura 3.4 Formulario Agregar Producto

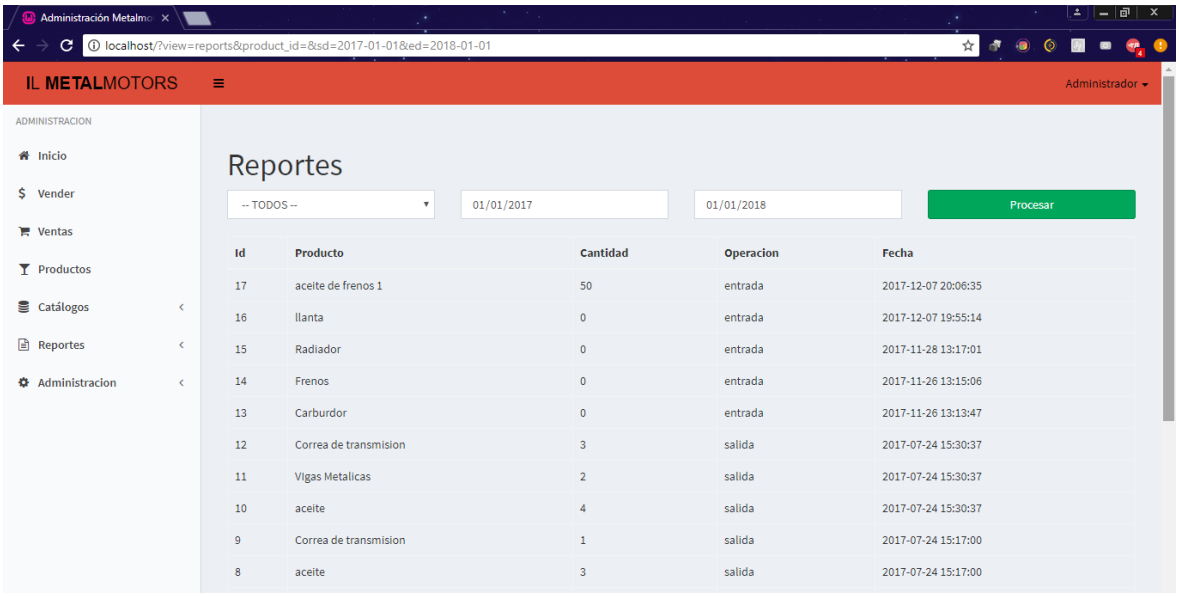


Fuente: Aplicación IL METALTOURS

Autor: Fernando Maila

El módulo Reportes permite realizar una consulta global de todos los productos que ingresados en un periodo determinado de tiempo como se puede observar en la Figura 3.5.

Figura 3.5 Generación de reporte de inventario



Fuente: Aplicación IL METALTOURS

Autor: Fernando Maila

3.2.1.4 Metodología para la aplicación de pruebas

La metodología de trabajo que se utilizará para ejecutar las pruebas en la aplicación web de la empresa “METALMOTORS S.A” se basa en establecer un escenario de prueba, descargar e instalar la herramienta Open Source seleccionada, definir casos de prueba, creación de scripts y la implementación de las pruebas. Los resultados obtenidos registran en una tabla en la que se generará gráficas y finalmente luego de un análisis obtener las conclusiones.

3.2.1.5 Cronograma de implementación de pruebas

La aplicación web “METALMOTORS S.A” será sometida a pruebas similares con diferentes cargas para cada caso de prueba, este proceso está programado en dos actividades, la primera es la creación de los scripts y la ejecución de las pruebas con distintas cargas.

El cronograma establecido para la implementación de pruebas se muestra en la Tabla 3.1.

Tabla 3.1 Cronograma de actividades para aplicación de pruebas

Actividades	Periodo de tiempo
Creación de Scripts	14 de agosto de 2017 – 19 de agosto de 2017
Ejecución primer caso de prueba	21 de agosto de 2017 – 25 de agosto de 2017
Ejecución segundo caso de prueba	28 de agosto de 2017 – 1 de septiembre de 2017
Ejecución tercer caso de prueba	4 de septiembre de 2017 – 8 de septiembre de 2017

Fuente: Fernando Maila

Autor: Fernando Maila

3.2.1.6 Responsable de las pruebas

El responsable de las pruebas es Edison Fernando Maila, estudiante egresado de Ingeniería en Sistemas Informáticos y de computación de la Escuela Politécnica Nacional.

3.2.1.8 Riesgos

El nivel del riesgo sobre la aplicación web a ser probada es bajo.

3.2.2 Análisis y diseño de pruebas

3.2.1 Instalación de herramientas Open Source seleccionadas

En la instalación de la herramienta se detalla donde descargar el instalador, los requisitos hasta su ejecución y finalmente es importante especificar las características del computador en el cual se ejecutarán las herramientas.

Las especificaciones de la máquina en la que se van a realizar las pruebas se detallan en la *Tabla 3.2.*

Tabla 3.2 Especificaciones del computador en el que se realizan las pruebas

Característica	Detalle
Tipo de computador	Laptop
Marca	HP
Modelo	Notebook HP 1000-1324LA
Procesador	Intel Core i5-3230M de tercera generación a 2,6 GHz, hasta 3,2 GHz con tecnología Turbo Boost
Memoria RAM	SDRAM DDR3 de 8 GB (1 DIMM)
Disco Duro	
Gráficos de video	Intel HD Graphics 4000 con memoria total de gráficos de hasta 1696 MB
Pantalla	Pantalla con retroiluminación LED de alta definición BrightView de 14 pulgadas en diagonal (1366 x 768)
Tarjeta de red	LAN Ethernet 10/100BASE-T (conector RJ-45)
Conectividad inalámbrica	WLAN 802.11b/g/n 1x1

Fuente: HP

Autor: Fernando Maila

3.2.3 Descarga de instaladores y requerimientos de instalación

La descarga del instalador se lo realiza en la página oficial de la aplicación. La aplicación tiene algunos requisitos para poder funcionar adecuadamente. A continuación, se presenta en la *Tabla 3.3* la herramienta con sus requerimientos para su instalación.

Tabla 3.3 Información de herramientas Open Source para su instalación

	JMeter
Página oficial de a aplicación	http://jmeter.apache.org/
Requerimientos	Java 8 o superior
Plataforma	Microsoft Windows, Linux

Fuente: Sitio oficial JMeter

Autor: Fernando Maila

La herramienta se la puede descargar en diferentes formatos dependiendo de la plataforma en la que se vayan a instalar, para este caso en particular las herramientas han sido descargadas en formato .rar por lo que una vez descomprimido se generan una carpeta con el contenido de la aplicación.
El ejecutable de la aplicación JMeter se encuentra en la carpeta /bin/ApacheJMeter.jar.

3.2.2.3 Define Caso de prueba

Los casos de prueba que se definen son tres, estos se detallan a continuación:

Caso de prueba 1

El usuario realiza se autentica para acceder al sistema.

Caso de prueba 2

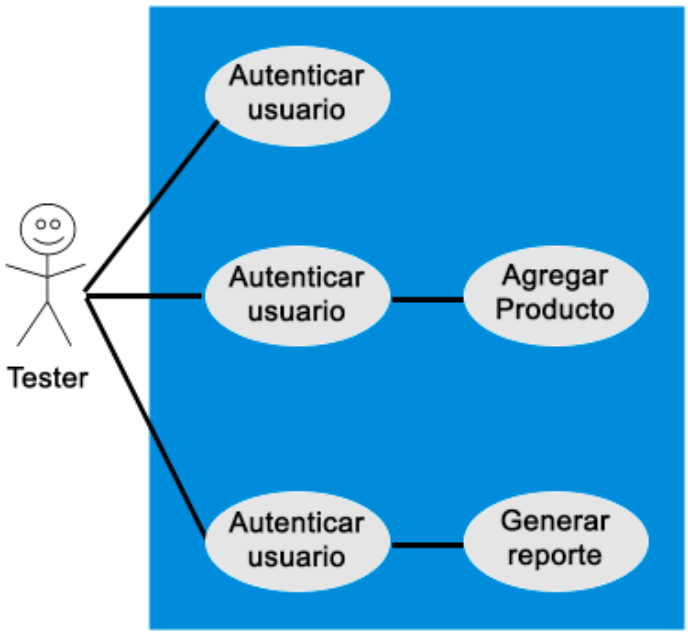
El usuario realiza se autentica para acceder al sistema, posteriormente agrega un nuevo producto.

Caso de prueba 3

El usuario realiza se autentica para acceder al sistema, el usuario genera un reporte del inventario.

Para los tres casos de prueba la autenticación se la aplica, con mas claridad se puede observar en la Figura 3.6.

Figura 3.6 Casos de prueba



Fuente: Fernando Maila

Autor: Fernando Maila

La autenticación se la aplica para los tres casos de prueba porque si el usuario no está autenticado no puede acceder a la aplicación.

3.2.3 Aplicación de herramientas

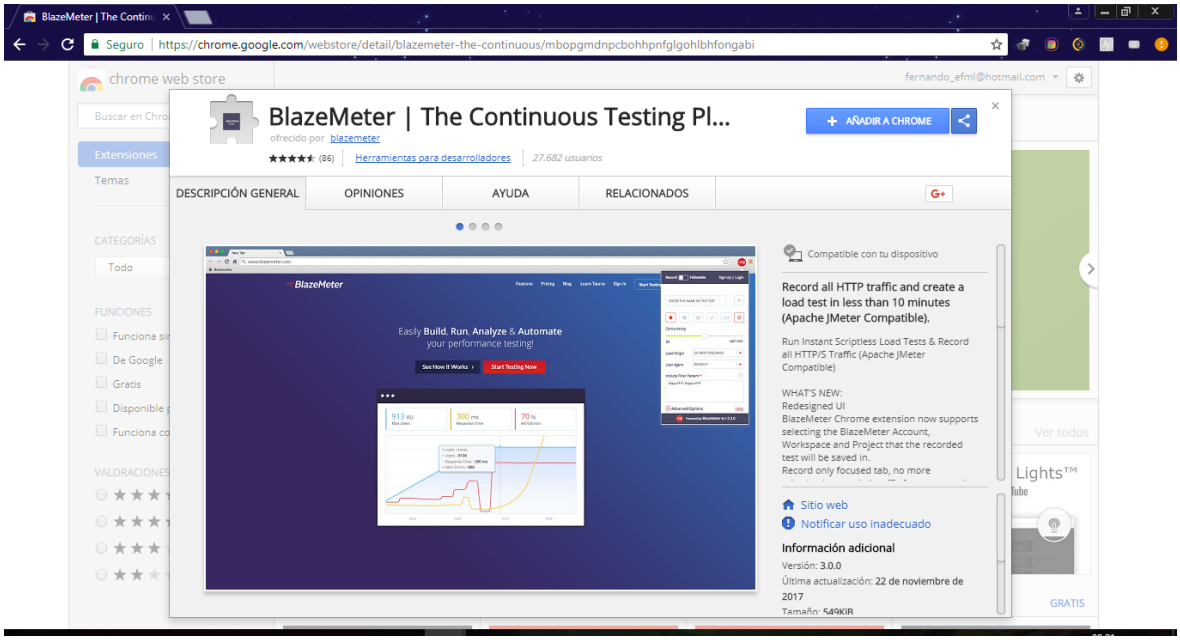
Para el proceso de aplicación de la herramienta Open Source, iniciar con la ejecución de la aplicación de web, para ello se usa un servidor para aplicaciones web que es XAMP, este es una distribución de Apache el cual permite la administración de la base de datos y la interpretación de lenguajes PHP y Perl. Una vez ejecutado el servidor con la aplicación web “METALMOTORS S.A” [10]

Una vez ejecutada la aplicación web de manera correcta, se procede a ejecución de la herramienta JMeter.

Uso de JMeter en el caso de estudio

Para proceder a realizar las pruebas a la aplicación web “METALMOTORS” con la herramienta JMeter es necesario generar los scripts, en este caso para el primer caso de prueba que es el de autenticación, para ello se descarga una extensión de Google Chrome que se llama BlazeMeter, la descarga se la realiza desde Web Store de Google como se muestra en la Figura 3.8. Esta extensión en resumen lo que hace es grabar el tráfico HTTP y crea un script compatible con la herramienta JMeter, esta se la exporta para posteriormente cargarla en JMeter. [11]

Figura 3.7 Extensión BlazeMeter en la Web Store de Google



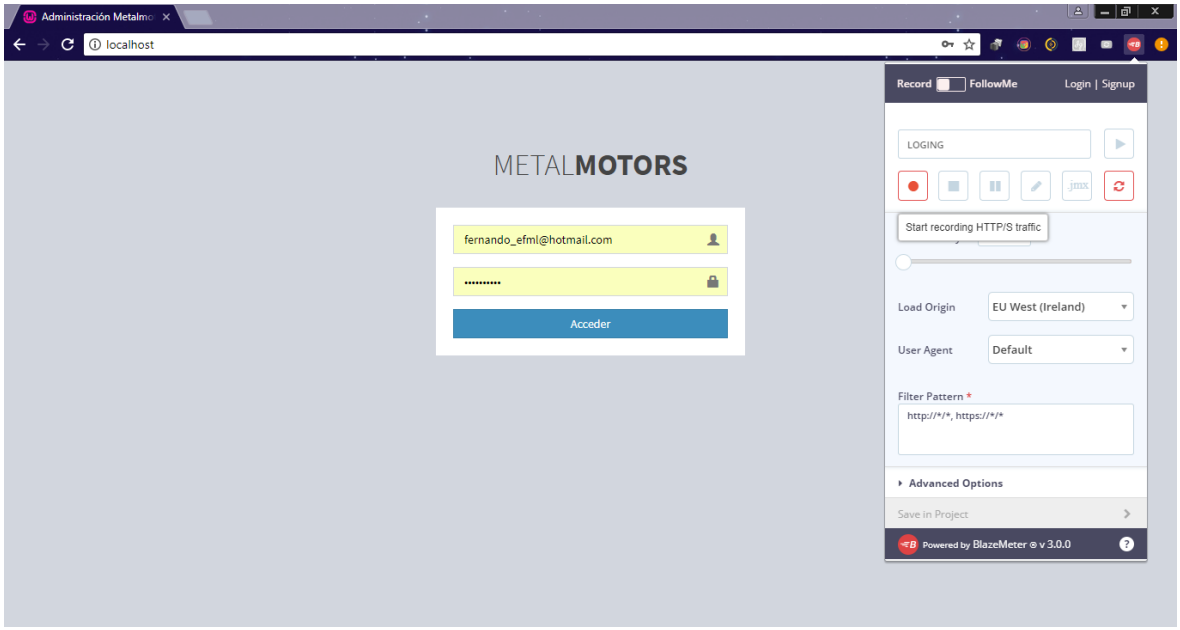
Fuente: Web Store de Google

Autor: Fernando Maila

Una vez instalada la extensión BlazeMeter en el navegador Google Chrome se procede a generar grabar procedimientos para cada uno de los casos de prueba.

Para este ejemplo que es el de autenticación, se inicia con dar clic en el botón de la extensión, se despliega una interfaz como se muestra en la Figura 3.9, ingresar un nombre y dar clic en el botón de *Rec* que guarde el registro de la autenticación.

Figura 3.8 Ejecución de la herramineta BlazeMeter en el caso de estudio

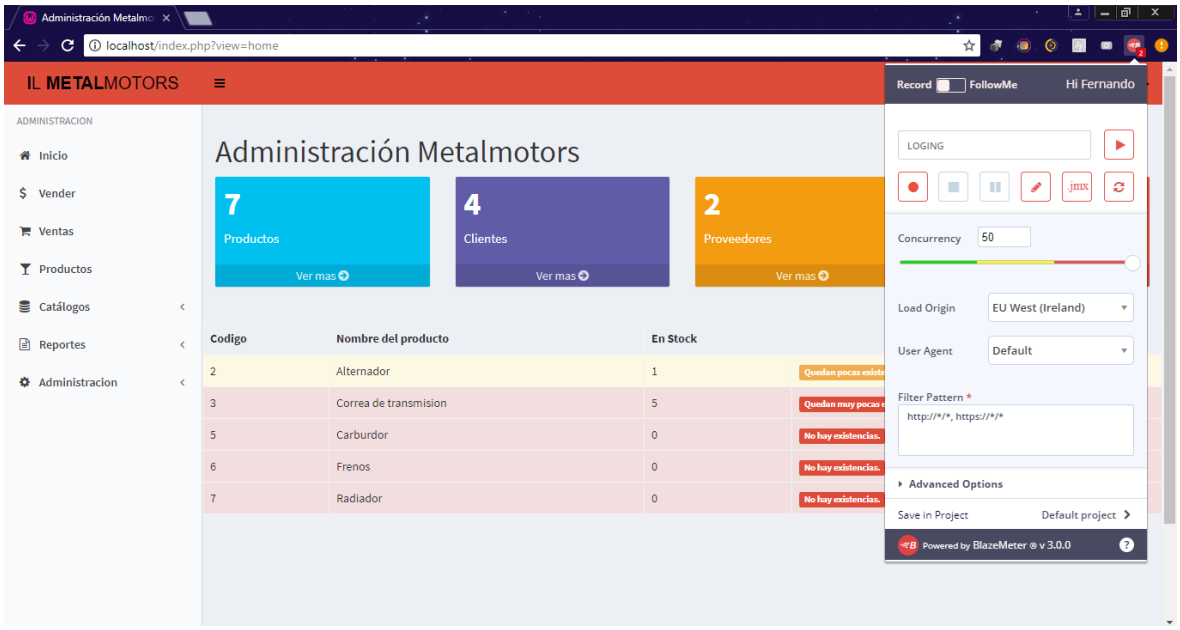


Fuente: Fernando Maila

Autor: Fernando Maila

Ya realizada la autenticación se procede a detener la extensión dando clic en el botón de *stop*. En la *Figura 3.10* se puede ver la aplicación en pleno funcionamiento.

Figura 3.9 BlazeMeter en funcionamiento en la aplicación del caso de estudio

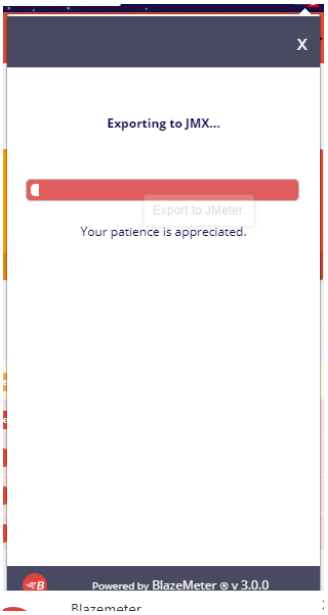


Fuente: Fernando Maila

Autor: Fernando Maila

Una vez detenida la extensión, se exporta el script generado para posteriormente abrirlo en la herramienta JMeter como se puede observar en la *Figura 3.11*.

Figura 3.10 Exportación del Script desde BlazeMeter

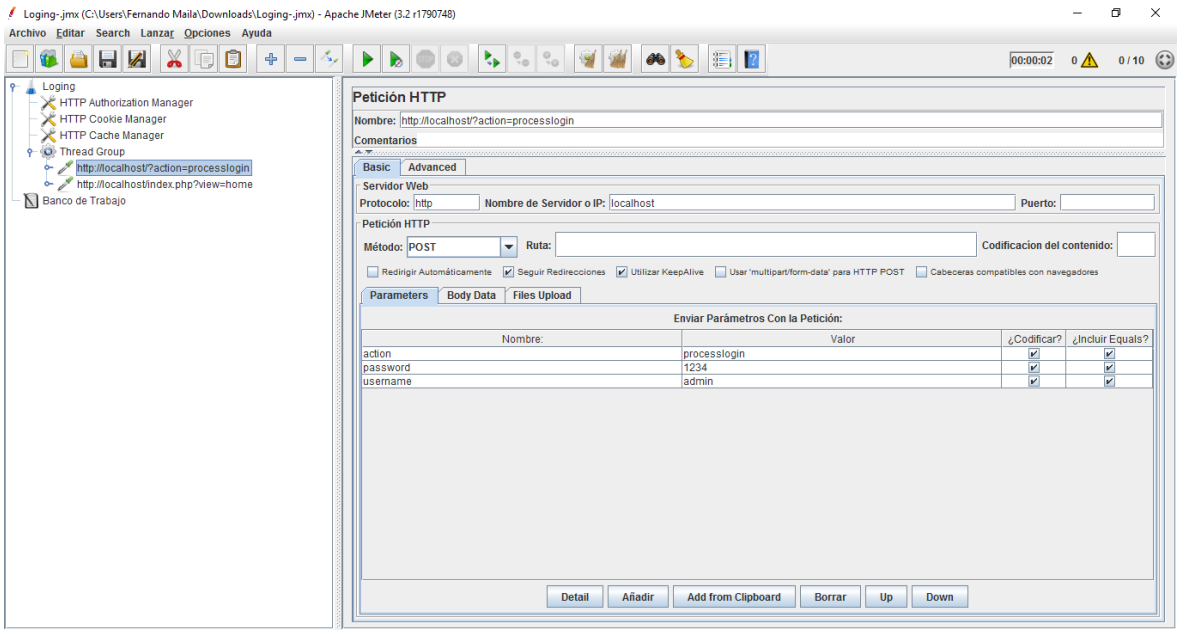


Fuente: Fernando Maila

Autor: Fernando Maila

Una vez exportado se procede abrir el script en JMeter, como se puede observar en la Figura 3.12 los parámetros de autenticación ya están configurados.

Figura 3.11 Script generado en BlazeMeter cargado en JMeter para el caso de estudio

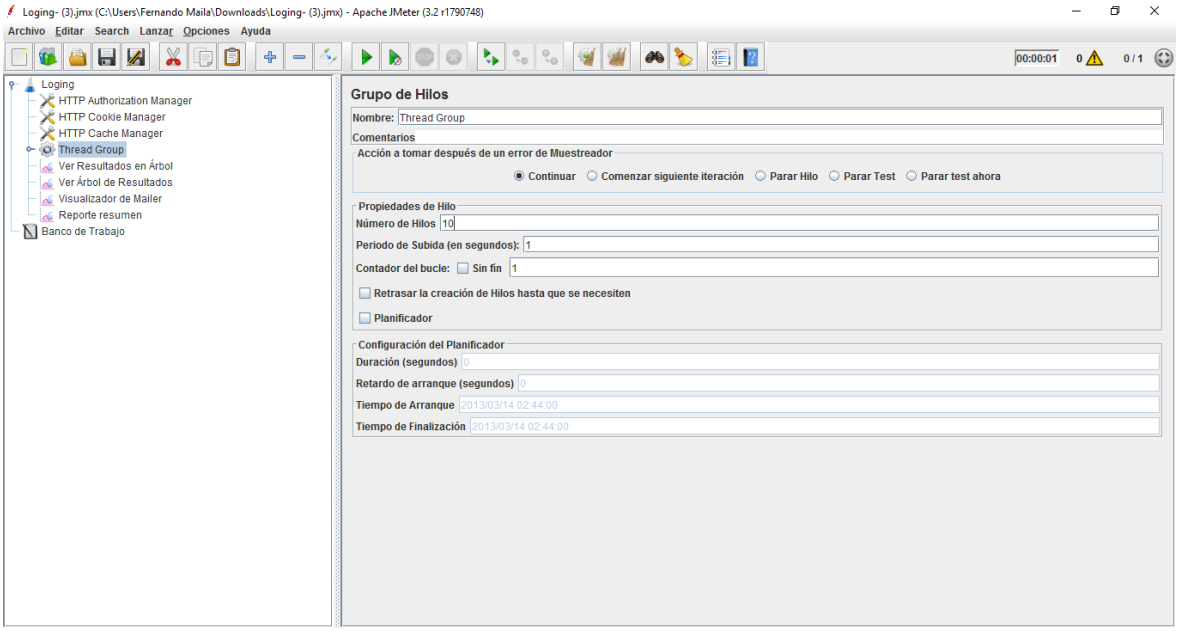


Fuente: Fernando Maila

Autor: Fernando Maila

Antes de iniciar la prueba se debe configurar el grupo de *threads* (hilos) con los que se va a ejecutar cada caso de prueba, además determina qué ocurre si se produce un error, ya sea porque la muestra falló o falló una aserción. Las posibles opciones en caso de presentarse un error son: continuar: ignore el error y continúe con la prueba. El grupo de threads tiene muchas opciones como se puede ver en la Figura 3.13 para poner a la herramienta en un punto extremo para las pruebas, el periodo de despliegue de los threads va a ser de 1 segundo.

Figura 3.12 JMeter, configuración grupo de *Threads*

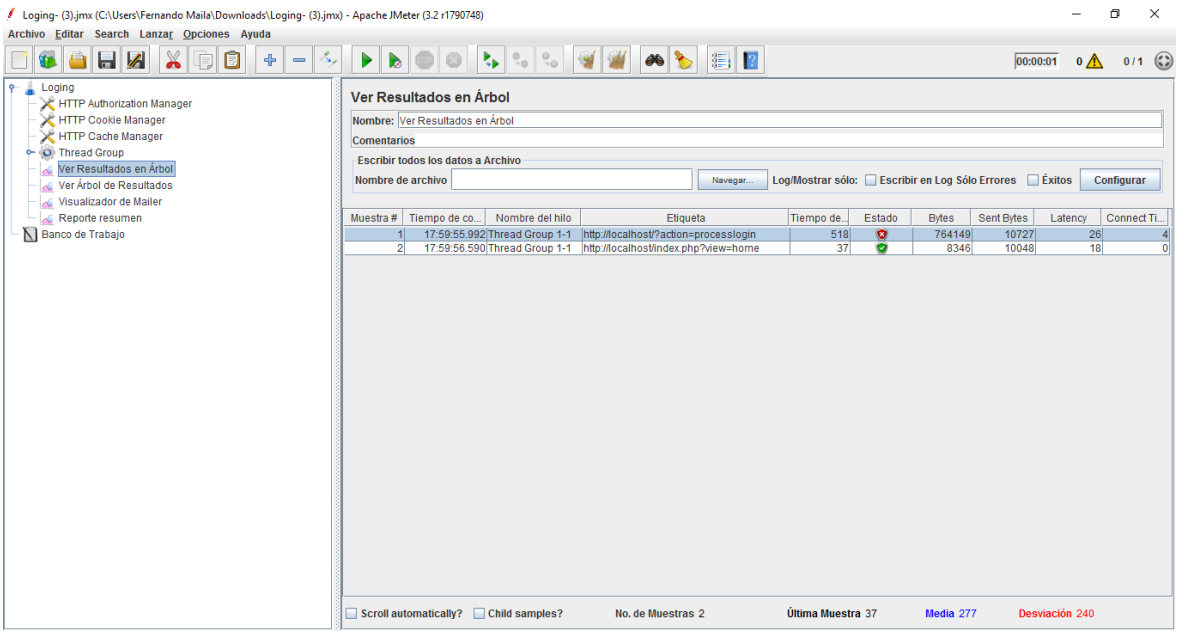


Fuente: Fernando Maila

Autor: Fernando Maila

En este ejemplo en la Figura 3.14 se puede observar la ejecución de una prueba con un solo *thread* si bien se puede observar un mensaje que parece ser error, es una alerta que hubo un problema para descargar un archivo .css. Es un error que en el proceso en muchos de los casos de prueba se presentan. Se pueden mostrar los resultados de muchas formas, en árboles, gráficos, resumen de resultados, etc. En este ejemplo se muestra los resultados en un árbol, en el detalle esta: números de muestra, tiempo de comienzo, nombre del thread (hilo), etiqueta, tiempo de muestra, bytes, bytes enviados, latencia y el tiempo de conexión.

Figura 3.13 JMteer, resultados de la ejecución de la prueba en árbol

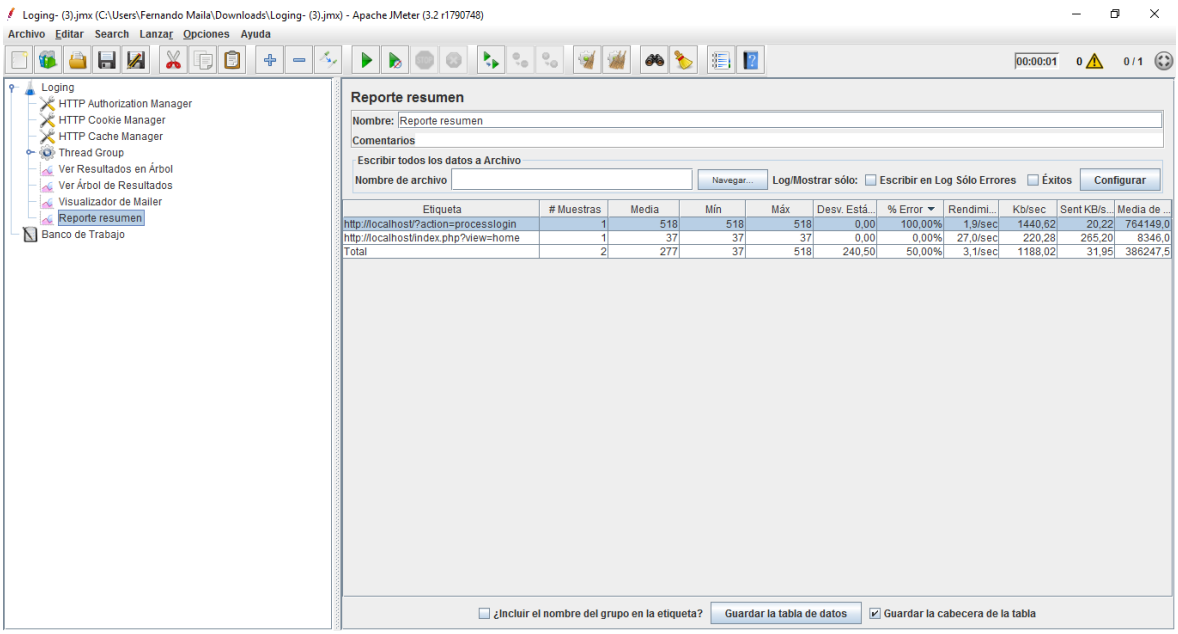


Fuente: Fernando Maila

Autor: Fernando Maila

En la *Figura 3.15* se puede revisar los resultados en un resumen. En este informe de resultados se puede observar el número de muestras, promedio, min, max, la desviación estándar, porcentaje de error, rendimiento, los Kb/s recibidos, Kb/s enviados y el promedio de Bytes.

Figura 3.14 Resumen de resultados obtenidos de un caso de prueba.



Fuente: Fernando Maila

Autor: Fernando Maila

3. RESULTADOS Y DISCUSIÓN

Las pruebas se las realizó aplicando la herramienta JMeter en diferentes casos de prueba con diferentes cargas. Los resultados obtenidos se presentan a continuación en la *Tabla 3.4*, donde se puede apreciar el número de threads aplicados, el caso de prueba, el tiempo de respuesta total, el tiempo promedio, el porcentaje de error y el rendimiento.

Tabla 3.4 Resultados con la herramienta JMeter

Herramienta	Jmeter						
# threads	1						
Métricas / Casos de Prueba	Tiempo de respuesta (ms)	Tiempo de espera (s)	Rendimiento (peticiones/min)	Número de peteciones (peticiones/s)	Número de accesos simultaneos (num accesos/s)	Sistemas de transmisión de ancho de banda (Kb/s)	%error
Login	763	1	5	0,5	1	21,04	50%
Ingresar Producto	647	38	9,2	0,15	0,02	20.13	16,67%
Inventario	633	20	11,8	0,2	0,05	37,9	25%

Herramienta	Jmeter						
# threads	10						
Métricas / Casos de Prueba	Tiempo de respuesta (ms)	Tiempo de espera (s)	Rendimiento (peticiones/min)	Número de peteciones (peticiones/s)	Número de accesos simultaneos (num accesos/s)	Sistemas de transmisión de ancho de banda (Kb/s)	%error
Login	206	3	5,5	6,66	3,33	2083,37	50%
Ingresar Producto	665	41	1,5	1,47	0,24	191,68	16,67%
Inventario	701	21	1,8	1,9	0,47	354,43	25%

Herramienta	Jmeter						
# threads	50						
Métricas / Casos de Prueba	Tiempo de respuesta (ms)	Tiempo de espera (s)	Rendimiento (peticiones/min)	Número de peteciones (peticiones/s)	Número de accesos simultaneos (num accesos/s)	Sistemas de transmisión de ancho de banda (Kb/s)	%error
Login	3237	49	2	2,04	1,02	757,89	50%
Ingresar Producto	664	138	2,2	2,17	0,36	283,08	16,67
Inventario	678	99	2	2,02	0,5	287,97	25%

Herramienta	Jmeter						
# threads	100						
Métricas / Casos de Prueba	Tiempo de respuesta (ms)	Tiempo de espera (s)	Rendimiento (peticiones/min)	Número de peteciones (peticiones/s)	Número de accesos simultaneos (num accesos/s)	Sistemas de transmisión de ancho de banda (Kb/s)	%error
Login	3745	91	*2,2*	2,19	1,09	819,86	60,50%
Ingresar Producto	664	273	*1.47*	3,29	0,38	194,38	29,56%
Inventario	761	189	*2,1*	2,11	0,52	409,08	42%

Herramienta	Jmeter						
# threads	200						
Métricas / Casos de Prueba	Tiempo de respuesta (ms)	Tiempo de espera (s)	Rendimiento (peticiones/min)	Número de peteciones (peticiones/s)	Número de accesos simultaneos (num accesos/s)	Sistemas de transmisión de ancho de banda (Kb/s)	%error
Login	908	81	*4,9*	4,93	2,46	1170,79	93,75%
Ingresar Producto	888	236	*5,1*	5,08	0,84	640,3	87,42
Inventario	1078	121	*4,9*	6,6	1,65	809,99	89,62%

Fuente: JMeter

Autor: Fernando Maila

De los datos obtenidos se puede tener como conclusiones que:

- En las pruebas realizadas con una carga mínima de uno, los tiempos de espera obtenidos son bajos, carcanos al segundo.
- A medida que la carga va en aumento (número de threads), los tiempos de respuesta también aumentan de manera significativa.
- La aplicación ““METALMOTORS S.A” con cargas mínimas no presenta errores, pero a medida que la carga aumenta, la aplicación también presenta mayor cantidad de errores.
- La aplicación responde hasta una carga de cincuenta, en las pruebas que se realizan con carga de cien y doscientos los errores aumentan considerablemente.

4. CONCLUSIONES

- Los conceptos usados en este documento son una guía para el lector para poder entender los resultados obtenidos, las métricas usas y la manera de calcularlas.

- Las herramientas Open Source para evaluar rendimiento de aplicaciones web son una buena alternativa frente al software propietario, en este trabajo se pudo obtener una lista de ocho herramientas para realizar pruebas de rendimiento.

- No es recomendable usar herramientas Open Source para evaluar fiabilidad, si bien existen las herramientas Open Source para pruebas de fiabilidad, de la lista obtenida las herramientas no cumplen con una documentación adecuada, un instalador o son antiguas lo cual no garantizaría tener un adecuado soporte en caso de presentar problemas en su ejecución.

- Si bien uno de los objetivos de este trabajo era tener mínimo tres herramientas que evalúen fiabilidad, en la lista premilitar se tiene cuatro herramientas, al no cumplir los criterios de selección estas fueron descartadas y finalmente sólo se obtuvo una, Tsung, que cubre la subcaracterística disponibilidad.

- Existen muchas herramientas Open Source pero si es importante discernir que herramienta cumple ciertos parámetros para poder decir “esta es una herramienta recomendada”, en este trabajo se inició con una lista preliminar de dieciocho herramientas pero luego de someterse a los filtros solo quedaron ocho.

- Al comparar una herramienta con otra se puede verificar la consistencia de datos, si bien las herramientas permiten calcular ciertas métricas de las características de fiabilidad y rendimiento, se pudo verificar que las herramientas pueden hacer esos cálculos de manera distinta como es el caso de JMeter y Gatling en la métrica de tiempo de respuesta.

- La herramienta JMeter es muy versátil, cuenta con una interfaz muy fácil de usar, cuenta con muchas alternativas para realizar pruebas de carga y permite apreciar los resultados en diferentes presentaciones, arboles de resultados, resultados en árbol, de manera individual por cada thread, en informe resumido, en gráficos, etc. Una herramienta recomendada para quienes necesiten realizar pruebas de carga.

- JMeter permite medir las métricas; tiempo de respuesta, tiempo de espera, y rendimiento de la subcaracterística comportamiento en el tiempo; número máximo de peticiones, número de accesos simultáneos y el sistema de transmisión de ancho de banda de la subcaracterística capacidad.

Bibliografía

[1] RAE, «Real Academia de la Lengua,» [En línea]. Available: http://dle.rae.es/?id=6nVpk8P|6nXVL1Z. [Último acceso: 23 marzo 2017].

[2] ISO/IEC FCD 25000:2004(E), «Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE,» nº 43, 30 noviembre 2014.

[3] ISO/IEC/IEEE 24765:2010(E), «Systems and software engineering - Vocabuary,» 2010.

[4] ISO/IEC WD 25023, «Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality,» nº 54, 2011.

[5] International Software Testing Qualifications Board, «ISTQB,» [En línea]. Available: http://www.istqb.org/about-as/vision-mission.html. [Último acceso: julio 2017].

[6] International Software Testing Qualifications Board, «Foundation level Syllabus,» 2011. [En línea]. Available: http://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html4. [Último acceso: mayo 2017].

[7] S. C. B. d. Souza, N. Anquetil y K. M. d. Oliveira, «Universidade Católica de Brasília, Brasilia, Brazil,» noviembre 2005. [En línea]. Available: http://dl.acm.org/citation.cfm?id=1085331. [Último acceso: mayo 2017].

[8] N. Jemutai Kipyegen y W. Korir, «International Journal of Computer Science Issues,» septiembre 2013. [En línea]. Available: https://ijcsi.org/papers/IJCSI-10-5-1-223-228.pdf. [Último acceso: 12 mayo 2017].

[9] B. Damian, «fwptt,» 2008. [En línea]. Available: http://fwptt.sourceforge.net/index.html. [Último acceso: 6 marzo 2017].

[10] Apache Friends, «Apache Friends,» [En línea]. Available: https://www.apachefriends.org/es/index.html. [Último acceso: 13 agosto 2017].

[11] Chrome Web Store, «BlazeMeter,» [En línea]. Available: https://chrome.google.com/webstore/detail/blazemeter-the-continuous/mbopgmdnpcbohhpnfglgohlbhfongabi. [Último acceso: 15 octubre 2017].

[12] ISO/IEC JTC1/SC7, «CD 25010.2, Software engineering-Software product Quality,» p. 42, 2008.

[13] B. Delbosc, «funkload,» 11 mayo 2015. [En línea]. Available: http://funkload.nuxeo.org/index.html. [Último acceso: 06 marzo 2017].

[14] CLIF, «CLIF,» 25 01 2017. [En línea]. Available: http://clif.ow2.org/. [Último acceso: 06 2017].

[15] Gatling Corp, «Gatling Load Testing,» 2017. [En línea]. Available: http://gatling.io/. [Último acceso: 06 2017].

[16] N. Niclausse, «A distributed load testing tool Tsung,» 02 05 2013. [En línea]. Available: http://tsung.erlang-projects.org/. [Último acceso: 29 06 2016].

[17] HP, «HP Ecuador,» [En línea]. Available: https://support.hp.com/ec-es/document/c03632068. [Último acceso: 13 07 2016].

[18] XCEPTANCE Software Technologies, «Xceptance LoadTest,» 15 junio 2017. [En línea]. Available: https://www.xceptance.com/en/. [Último acceso: 23 julio 2017].

[19] Xceptance GmbH, «FIrefox Add-ons,» octubre 2017. [En línea]. Available: https://addons.mozilla.org/es/firefox/addon/xceptance-script-developer/.