Universidad Mariano Gálvez de Guatemala

Ingeniería en Sistemas

Curso: Aseguramiento De La Calidad De Software

Ing. Carmelo Estuardo Mayén Monterroso

Contenido:

Investigación y Documentación de Herramientas para Pruebas de Software Esenciales

Estudiante:

Jeremy Saul Rodriguez Garcia

Carnet:

1790-20-20725

11 de Agosto de 2025

Tipos de Pruebas a Investigar:

1. Pruebas Funcionales.
2. Pruebas de Rendimiento.
3. Pruebas de Seguridad.

Introducción

En este trabajo vamos a investigar y documentar algunas herramientas importantes que se utilizan para hacer pruebas de software. La idea es entender para qué sirven y cómo ayudan a que las aplicaciones funcionen bien antes de que lleguen a los usuarios.

Las pruebas de software son muy importantes porque nos permiten detectar errores y problemas a tiempo. Así evitamos que los usuarios tengan una mala experiencia o que la aplicación falle en momentos importantes. Es como revisar un trabajo antes de entregarlo para asegurarnos de que todo esté correcto.

Dentro de estas pruebas, hay diferentes tipos que cumplen funciones específicas. Las **pruebas funcionales** revisan que el sistema haga lo que se espera. Las **pruebas de rendimiento** se enfocan en ver qué tan rápido y estable es el software cuando muchas personas lo usan al mismo tiempo. Y las **pruebas de seguridad** ayudan a encontrar y prevenir fallas que podrían ser aprovechadas por personas con malas intenciones.

Conocer estas pruebas y las herramientas para realizarlas es clave para crear aplicaciones de calidad, confiables y seguras.

1. **Pruebas Funcionales.**

- **¿Qué son las pruebas funcionales?**

Las pruebas funcionales son un tipo de prueba de software que evalúan las funcionalidades específicas de una aplicación según los requerimientos del cliente o del negocio. En otras palabras, se encargan de verificar si el sistema realiza lo que debe hacer, sin enfocarse en el "cómo" lo hace (a diferencia de las pruebas estructurales).

Este tipo de pruebas se centran en validar:

- Entradas y salidas del sistema
- Flujos de usuario completos
- Reacciones ante errores
- Cumplimiento de reglas de negocio

Por ejemplo, en una aplicación bancaria, una prueba funcional puede verificar que al hacer clic en "Transferir", el sistema realice correctamente la transacción y actualice los saldos.

Tipos de pruebas funcionales:

- **Pruebas de humo (Smoke Testing):** Verifican funciones básicas en un entorno inicial.
- **Pruebas de regresión**: Aseguran que las nuevas funcionalidades no afecten el software existente.
- **Pruebas de integración**: Evalúan la interacción entre módulos del sistema.
- **Pruebas de aceptación** del usuario (UAT): Permiten que usuarios reales validen el sistema según sus expectativas.
- **Pruebas de caja negra**: Evalúan entradas y salidas sin considerar la estructura interna del código.

¿Cuándo se deben realizar?

Desde el inicio del ciclo de vida del software. Lo ideal es incorporar pruebas funcionales desde etapas tempranas (por ejemplo, con pruebas de aceptación a historias de usuario) e integrarlas en cada iteración del proyecto si se trabaja bajo metodologías ágiles.

En modelos DevOps o Continuous Delivery, las pruebas funcionales se automatizan y se integran en los pipelines CI/CD para evaluar cambios en tiempo real.

- **Herramientas comunes para pruebas funcionales**

Selenium – Automatización de pruebas funcionales en navegadores.

Postman – Para validar APIs funcionales.

- **Características de Selenium IDE**

Selenium IDE está repleto de funciones que simplifican y optimizan las pruebas web. Es ideal tanto para principiantes como para testers experimentados. Su función de grabación y reproducción, la sencilla gestión de casos de prueba y la compatibilidad con múltiples lenguajes de programación lo distinguen como una herramienta versátil dentro de la suite Selenium

Características más Destacadas:

1. Compatibilidad entre navegadores
2. Pruebas funcionales robustas
3. Mejor manejo de las funciones de la interfaz de usuario
4. Mejor soporte para código incrustado
5. Ejecución desde scripts de compilación de CI

- **Características de Postman**

Postman es una herramienta de colaboración y desarrollo que permite a los desarrolladores interactuar y probar el funcionamiento de servicios web y aplicaciones. proporcionando una interfaz gráfica intuitiva y fácil de usar para enviar solicitudes a servidores web y recibir las respuestas correspondientes

Características mas Destacadas:

- Envío de solicitudes. Permite enviar solicitudes GET, POST, PUT, DELETE y otros métodos HTTP a una API especificando los parámetros, encabezados y cuerpo de la solicitud.
- Gestión de entornos. Facilita la configuración para diferentes entornos (por ejemplo, desarrollo, prueba, producción) y el cambio sencillo entre ellos (para realizar pruebas y desarrollo en diferentes contextos).
- Colecciones de solicitudes. Agrupa las solicitudes relacionadas en colecciones, lo que facilita la organización y ejecución de pruebas automatizadas.
- Pruebas automatizadas. Es ideal para crear y ejecutar pruebas automatizadas para verificar el comportamiento de una API (detectar errores e incrementar la calidad del software).

Ventajas y desventajas de Selenium:

Ventajas	Desventajas
Código abierto y gratuito	Curva de aprendizaje pronunciada
Admite varios idiomas	Requiere conocimiento de lenguajes de programación
Capacidades de prueba entre navegadores	Sin soporte nativo para aplicaciones móviles o de escritorio
Comunidad de usuarios grande y activa	Funciones de informes limitadas
Altamente flexible y personalizable	Sin reconocimiento de objetos ni capacidades de IA
Se integra con las canalizaciones de CI/CD	Solo automatización del navegador, no un marco de prueba completo

Ventajas de usar Postman

1. Facilidad de uso: La interfaz es sencilla, lo que facilita a los principiantes comenzar.
2. Colaboración: Excelentes funciones para la colaboración y el intercambio en equipo.
3. Ligero: Requiere menos recursos del sistema en comparación con SoapUI.

Desventajas de usar Postman

1. Soporte de protocolos limitado: Se centra principalmente en las API REST, con soporte limitado para SOAP y otros protocolos.
2. Menos completo: Carece de algunas de las funciones de prueba avanzadas que ofrece SoapUI.

2. Pruebas de Rendimiento

¿Que son las pruebas de performance?

Las pruebas de performance pretenden medir la velocidad, los tiempos de respuesta, el uso de recursos y la escalabilidad de un sistema bajo determinada cantidad de carga, a modo de detectar cuellos de botella en el sistema bajo prueba.

¿Cuáles son los tipos de pruebas de performance?

Dentro de las pruebas de performance existen ciertos subtipos de ejecuciones, que nos permiten medir el comportamiento del sistema bajo diferentes circunstancias, por ejemplo:

- **Pruebas de carga**: Son pruebas que buscan medir el comportamiento del sistema bajo cargas poco usuales (pero esperadas) de usuarios, como por ejemplo cuando se abren inscripciones para un curso en línea con cupos limitados en una universidad o cuando se vence el plazo para pagar un servicio. En esos momentos se espera que mucha más gente de lo habitual ingrese al sistema y esto puede provocar enlentecimiento o caídas del mismo.
- **Pruebas de estrés:** A diferencia de las pruebas de carga, las pruebas de estrés buscan sumar carga al sistema de manera paulatina para identificar el punto de ruptura del mismo, es decir, hasta cuanta carga puede soportar un sistema sin romperse, o cuanto tiempo puede funcionar al límite de carga soportada de manera confiable.
- **Pruebas de escalabilidad**: Estas pruebas buscan ver como el sistema escala al tener que trabajar con más carga, por ejemplo, en un sistema que utilice AWS busca medir como se crean nuevos contenedores al aumentar la carga.

- **Herramientas para pruebas de performance**

Para poder hacer estas mediciones contamos con muchas herramientas de acceso libre que podemos utilizar, muchas de ellas online por lo que nos facilitan el trabajo cuando queremos tener un resultado rápido, y otras más configurable que nos permiten tener las pruebas integradas en un sistema de integración continua. Las herramientas que más utilizo son:

- Apache JMeter LoadRunner Blazemeter
- LoadImpact Testing Anywhere CloudTest

- **Características de Apache Jmeter**

Apache JMeter es una herramienta de pruebas de rendimiento de código abierto ampliamente utilizada para pruebas de carga de aplicaciones web, API y otros servicios. Originalmente desarrollado para probar aplicaciones web, JMeter se ha convertido en una herramienta integral para probar varios protocolos que incluyen HTTP, FTP, JDBC y SOAP.

Características clave de JMeter:

- **Código abierto y gratuito:** JMeter es de uso completamente gratuito, lo que lo convierte en una opción atractiva para pequeñas empresas y nuevas empresas.
- Soporta múltiples protocolos: Puede probar aplicaciones web, API, rendimiento de bases de datos, servidores FTP, servicios SOAP y más.
- **Amplio soporte de complementos:** La comunidad de JMeter ofrece numerosos plugins para ampliar sus capacidades.
- **Multiplataforma:** Dado que está escrito en Java, JMeter se ejecuta en cualquier sistema operativo.
- **Escalabilidad:** JMeter se puede integrar con servicios basados en la nube para pruebas de carga distribuidas.
- **Fácil integración**: Funciona con canalizaciones de CI/CD para permitir pruebas de rendimiento continuas.

Ventajas de JMeter:

1. Gratis y de código abierto
2. Gran apoyo de la comunidad
3. Fácil de configurar y usar
4. Admite secuencias de comandos con Beanshell, Groovy y Java
5. Altamente personalizable a través de plugins

Contras de JMeter:

1. Informes integrados limitados en comparación con LoadRunner
2. Puede consumir muchos recursos para pruebas a gran escala
3. La interfaz de usuario no es tan fácil de usar como las herramientas comerciales
4. Requiere más secuencias de comandos manuales para escenarios de prueba complejos

¿Qué es LoadRunner?

Micro Focus LoadRunner es una herramienta de pruebas de rendimiento comercial diseñada para aplicaciones de nivel empresarial. Es una de las herramientas más poderosas disponibles para simular miles o incluso millones de usuarios virtuales.

LoadRunner es conocido por sus capacidades avanzadas de scripting, que permiten a los evaluadores crear escenarios de prueba realistas y complejos. Con su Virtual User Generator (VuGen), los usuarios pueden crear scripts de casos de prueba en C y ejecutar pruebas de rendimiento a gran escala. Proporciona información detallada sobre el rendimiento del sistema para ayudar a las organizaciones a identificar cuellos de botella y optimizar sus aplicaciones antes de la implementación.

Características clave de LoadRunner:

- Soporte de protocolo: Admite más de 50 protocolos, incluidas aplicaciones web, móviles, de bases de datos y ERP.
- Secuencias de comandos avanzadas con VuGen: LoadRunner utiliza Virtual User Generator (VuGen) para crear scripts de prueba complejos con scripts basados en C.
- Analíticas potentes: Ofrece métricas detalladas de rendimiento, tiempos de respuesta de transacciones y detección de cuellos de botella.
- Herramientas de monitoreo integradas: Proporciona monitoreo en tiempo real e integración con herramientas APM.
- Soporte de pruebas en la nube: LoadRunner puede simular pruebas en entornos de nube.

Ventajas de LoadRunner:

1. Compatible con una amplia gama de protocolos
2. Análisis e informes sólidos
3. Gestión eficiente de los recursos, incluso bajo cargas elevadas
4. Supervisión del rendimiento en tiempo real
5. Soporte de nivel empresarial

Desventajas de LoadRunner:

1. Altos costos de licencia
2. Curva de aprendizaje empinada, especialmente para principiantes
3. Requisitos pesados del sistema
4. Requiere experiencia en scripting (el scripting basado en C puede ser complejo para los nuevos usuarios)

- **Pruebas de seguridad**

¿Qué son las pruebas de seguridad?

Identifica puntos débiles como vulnerabilidades, amenazas y riesgos en un sistema. Su objetivo es garantizar la seguridad de las aplicaciones informáticas. Además, comprueba que los datos sensibles no están expuestos a accesos no autorizados.

Las pruebas de seguridad de las aplicaciones comprueban la capacidad de una aplicación para salvaguardar los datos. También para mantener la confidencialidad, integridad y disponibilidad. Garantiza que se siguen los mecanismos adecuados de autenticación, autorización y no repudio.

Ejemplo de vulnerabilidad: Inyección SQL

Los atacantes manipulan las consultas a la base de datos de la aplicación. Lo que significa que pueden acceder a datos que no deberían. Por ejemplo, contraseñas, datos financieros, información personal, etc. Incluso pueden cambiar o borrar estos datos. Esto afecta al funcionamiento de la aplicación o a lo que muestra.

Para comprobar las vulnerabilidades de inyección SQL:

1. Introducir sentencias SQL maliciosas en los campos de la aplicación.
2. Observa el comportamiento del sistema.
3. Verificar si la base de datos o la información sensible está fuera.

- Herramientas para pruebas de seguridad.

1. **SonarQube:** Detecta las vulnerabilidades del código y aplica normas de calidad.
2. **OWASP ZAP**: Es una herramienta de código abierto para pruebas de seguridad de aplicaciones web

- **¿Qué es SonarQube?**

SonarQube es una plataforma web que se utiliza para analizar y cuantificar la calidad del código fuente. Es una herramienta de código abierto que provee varias ediciones dependiendo de la necesidad de los usuarios.

Aunque SonarQube está escrita en Java, permite analizar y administrar el código de muchos lenguajes de programación como C/C++, PL/SQL, Javascript, Typescript, entre muchos otros, usando complementos que permiten ampliar las funcionalidades de SonarQube. Actualmente existen más de 50 complementos disponibles.

- **Caracteristicas de SonarQube**

1. Arquitectura y diseño.
2. Código duplicado.
3. Errores potenciales.
4. Complejidad del código.
5. Estándares de codificación.
6. Comentarios.
7. Pruebas unitarias.

SonarQube usa los archivos del código fuente para realizar el análisis. Además, calcula un conjunto de métricas que son almacenadas en una base de datos para posteriormente mostrarlas en la interfaz web provista.

- **Ventajas de Utilizar SonarQube**

1- Detecta y notifica problemas

SonarQube detecta errores en el código automáticamente y alerta a los desarrolladores para que los incidentes sean corregidos antes de realizar el despliegue en ambientes de producción.

2- Aumenta la productividad

SonarQube aumenta la productividad al permitir a los equipos de desarrollo detectar y corregir la duplicidad y redundancia de código.

3- Optimiza la calidad

SonarQube puede funcionar como un analista multidimensional e informa sobre las siete secciones de la calidad del código mencionadas anteriormente.

4- Incrementa las habilidades del desarrollador

Una de las grandes ventajas de SonarQube es su adopción, pues los equipos de desarrollo pueden incorporarlo rápidamente.

5- Escala con las necesidades del negocio

SonarQube está diseñado para adaptarse a las necesidades del negocio. Aún no se ha descubierto ningún límite para su escalabilidad.

- **¿Qué es OWASP ZAP?**

OWASP ZAP (Zed Attack Proxy) es una herramienta de seguridad diseñada para detectar y solucionar vulnerabilidades en aplicaciones web.

Y lo mejor es que es de código abierto y fácil de usar, incluso si no eres un experto en ciberseguridad.

- **Caracteristicas de OWASP ZAP**

1. Escaneo de vulnerabilidades: Detecta problemas como inyecciones SQL y fallos de seguridad en cookies.
2. Automatización de seguridad: Puedes integrarlo en el desarrollo de software para detectar problemas antes de lanzar una actualización.
3. Protección de GLPI y aplicaciones internas: Asegura que tu plataforma de gestión de activos no tenga brechas de seguridad.
4. Proxy Interceptador: Te permite analizar y modificar el tráfico para pruebas más avanzadas.
5. Informes detallados: Te muestra qué problemas existen y cómo solucionarlos.

Ventajas de OWASP ZAP

1. **Gratis y de código abierto**: No cuesta nada y puedes modificarlo según lo que necesites.
2. **Fácil de usar para principiantes**: Con una interfaz gráfica clara, ideal para testers que apenas empiezan en seguridad.

3. **Escaneo activo y pasivo**: Detecta problemas sin interferir en el funcionamiento (pasivo) y también los ataca (activo) para ver si son vulnerables.
4. **Proxy interceptador**: Puedes inspeccionar y modificar tráfico web en tiempo real, como si fuera tu "espía" entre el navegador y el servidor.
5. **Personalizable y extensible**: Tiene complementos que puedes instalar para añadir funciones (como detección de CSRF, autenticación avanzada, etc.), y también admite scripts.

Desventajas de OWASP ZAP

1. **Menos intuitivo en la configuración avanzada**: Aunque es fácil empezar, ponerse serio con reglas y scripts puede resultar un poco técnico.
2. **Menor soporte en empresas grandes**: No tiene soporte de pago ni garantías comerciales como otras herramientas corporativas (por ejemplo, Burp Suite Enterprise).
3. **Limitado en análisis de APIs modernas o aplicaciones móviles**: Si bien hay complementos, no es tan directo ni especializado como otras soluciones diseñadas para APIs o apps móviles.
4. **Puede dar falsos positivos**: Algunas alertas pueden no ser realmente vulnerabilidades, es necesario revisarlas manualmente.

Conclusiones

Resumen de los hallazgos de la investigación

Las pruebas funcionales aseguran que el software haga lo que debe y que el usuario pueda usarlo sin problemas. Las pruebas de rendimiento ayudan a saber si el sistema funciona rápido y estable cuando muchas personas lo usan a la vez. Las pruebas de seguridad buscan proteger los datos y evitar que personas no autorizadas accedan al sistema o lo dañen.

Cada tipo de prueba cumple un papel importante para que el software sea útil, rápido y seguro.

Importancia de la elección de las herramientas adecuadas para el éxito del proyecto de software

Elegir las herramientas correctas para hacer las pruebas facilita mucho el trabajo, ahorra tiempo y ayuda a encontrar errores antes de que el software sea usado por los clientes. Además, tener buenas herramientas permite que todo el equipo pueda trabajar mejor y coordinarse para entregar un producto de calidad.

Sin las herramientas adecuadas, las pruebas pueden ser lentas, incompletas o poco claras, lo que pone en riesgo el éxito del proyecto.

Reflexión sobre el impacto de estas tres pruebas clave en la calidad y seguridad del software

Hacer pruebas funcionales, de rendimiento y de seguridad es como cuidar un coche antes de usarlo: se revisa que todo funcione, que aguante la carretera y que sea seguro para sus pasajeros. Si estas pruebas no se hacen bien, el software puede fallar, ser lento o estar expuesto a riesgos. Por eso, estas pruebas ayudan a que el software sea confiable, rápido y seguro para quienes lo usan, protegiendo tanto a los usuarios como a la empresa que lo desarrolla.

Recomendaciones

Selección de la herramienta adecuada

1- Es importante definir primero el tipo de prueba que se va a realizar: funcional, rendimiento o seguridad.
2- Se deben elegir herramientas acordes al tamaño y la complejidad del proyecto.
3- Es recomendable considerar el nivel de experiencia del equipo y optar por herramientas fáciles si el equipo es principiante.
4- La herramienta debe ser compatible con las tecnologías usadas en el desarrollo.
5- También es fundamental que la herramienta pueda integrarse con los procesos de desarrollo existentes.

Integración de las pruebas en el ciclo de desarrollo

1- Las pruebas deben iniciarse desde las primeras etapas del proyecto para detectar errores tempranamente.
2- Es recomendable automatizar las pruebas para facilitar su ejecución frecuente y constante.
3- Las pruebas deben integrarse en los procesos de desarrollo y entrega para validar automáticamente cada cambio.
4- Se debe fomentar la participación de todo el equipo en las pruebas para mejorar la calidad.
5- Finalmente, es necesario actualizar periódicamente las pruebas y herramientas según avance el proyecto.

Bibliography

- ¿Que son las pruebas Funcionales? - Definición

https://mtpinternational.mx/que-son-las-pruebas-funcionales-y-por-que-son-criticas-para-la-calidad-del-software

- Características de las Herramientas.

Selenium: https://www-browserstack-com.translate.goog/guide/features-of-selenium-ide?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Post Man: https://formadoresit.es/que-es-postman-cuales-son-sus-principales-ventajas

- Ventajas y Desventajas

Selenium: https://apidog.com/es/blog/best-automation-testing-tools-3

Postman: https://apidog.com/es/blog/soapui-vs-postman-4

- ¿Que son las pruebas de rendimiento?

https://elblogdesanti.com/ciencia-y-tecnologia/testing-automatizado/testing-de-performance

- Caracteristicas de Herramientas. (Apache Jmeter & LoadRunner) + Ventajas y Desventajas

https://www.loadview-testing.com/es/blog/jmeter-vs-loadrunner-para-pruebas-de-rendimiento-cual-es-mejor-y-por-que

- ¿Que son las pruebas de Seguridad?

https://powerdmarc.com/es/security-testing

- Caracteristicas de las Herramientas.

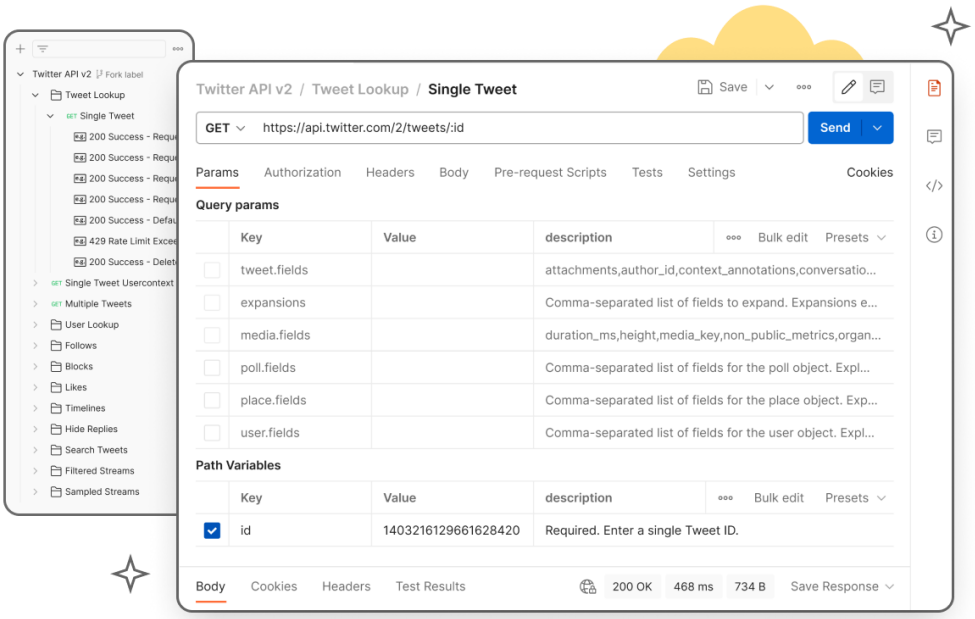
OWASP ZAP: https://casystem.com.mx/blog/protege-tus-aplicaciones-web-con-software-libre-owasp-zap?utm_

SonarQube: https://www.pragma.co/es/blog/conoce-las-8-ventajas-de-sonarqube

- **Anexos**

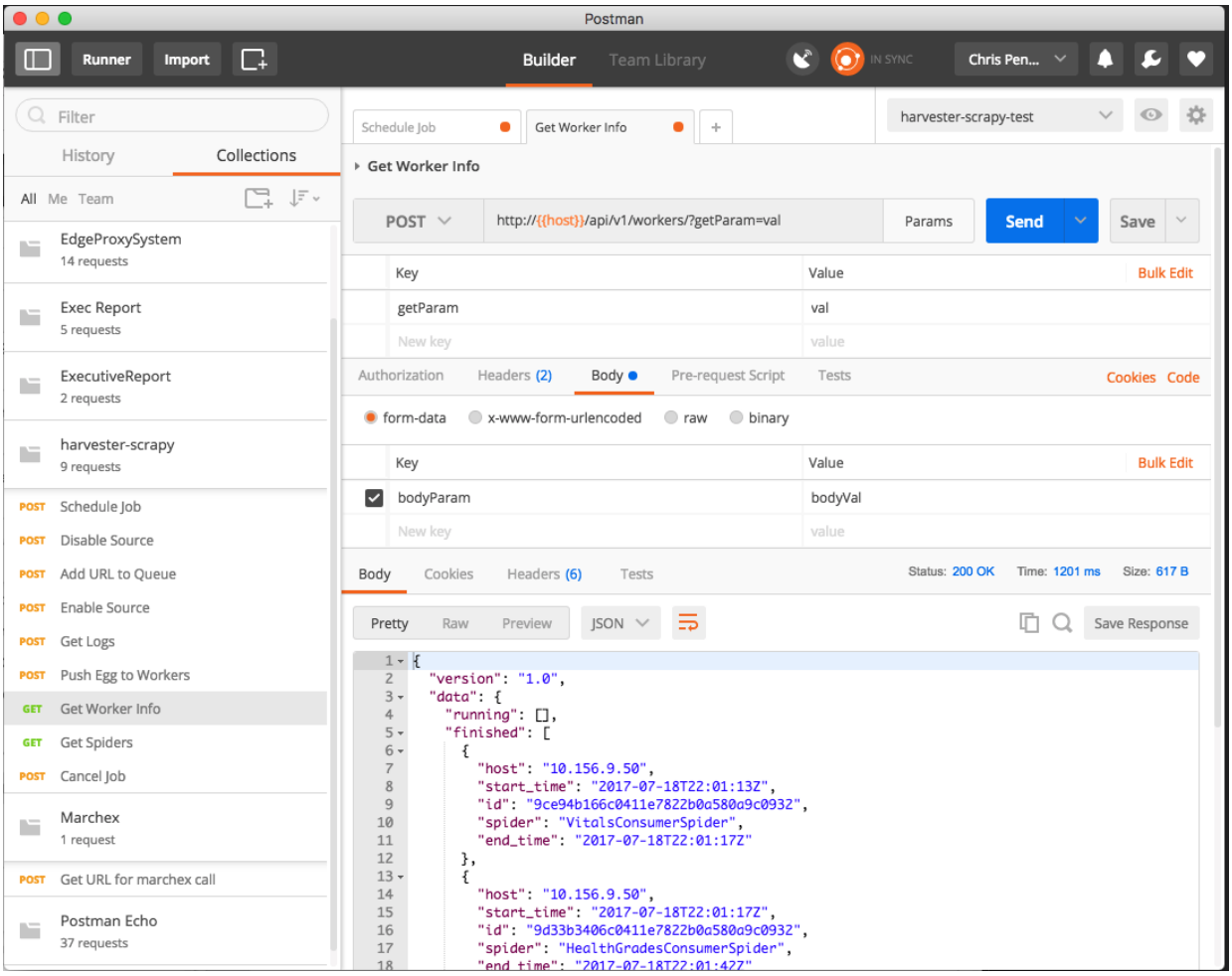
1- Postman

Captura #1



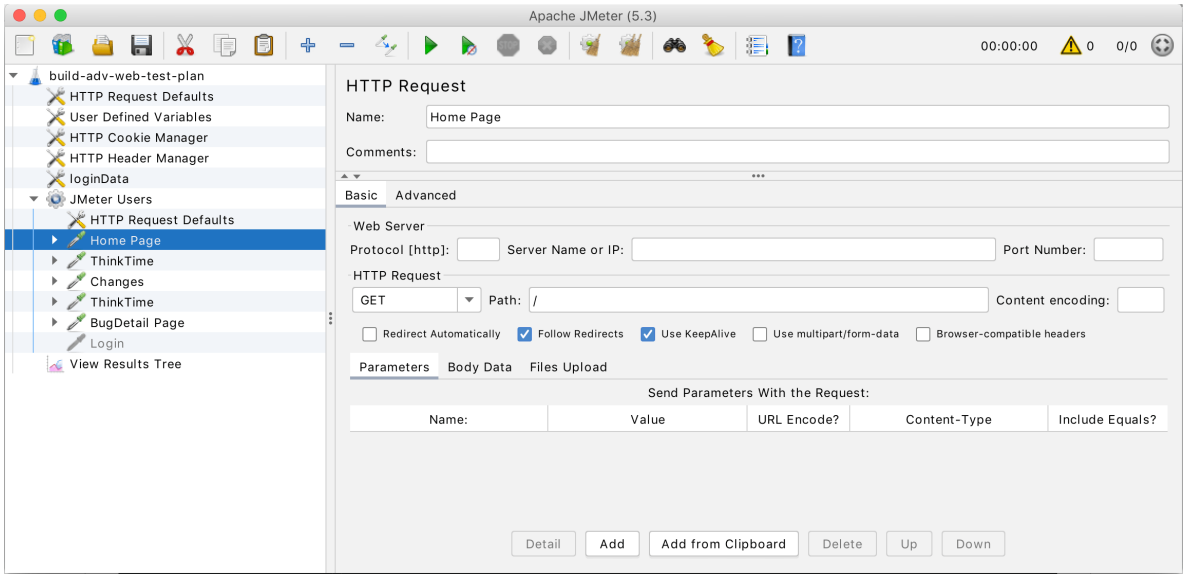
Descripción: Muestra la interfaz principal de Postman, con el panel para ingresar la URL, seleccionar el método HTTP, y un menú lateral para manejar colecciones y entornos. Ideal para ilustrar cómo se envían peticiones y organizan pruebas.

Captura #2

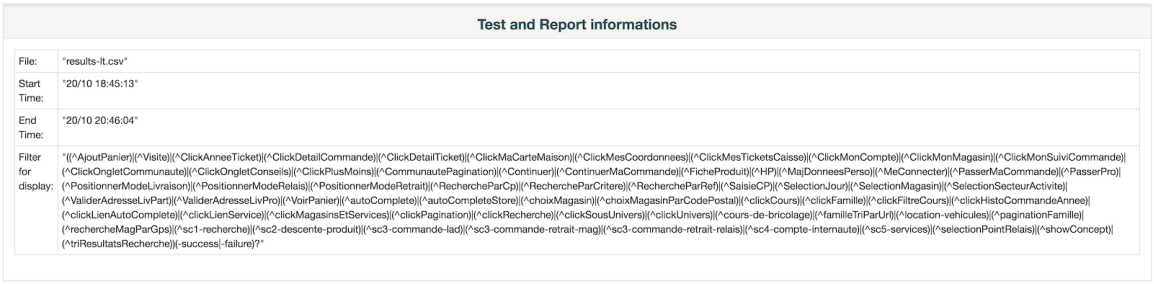


Descripción: Una vista clara del editor de peticiones, donde se ve el cuerpo de respuesta en formato JSON y opciones para editar headers, parámetros y scripts. Representa el flujo de pruebas funcionales.

2- Apache Jmeter – Captura #1

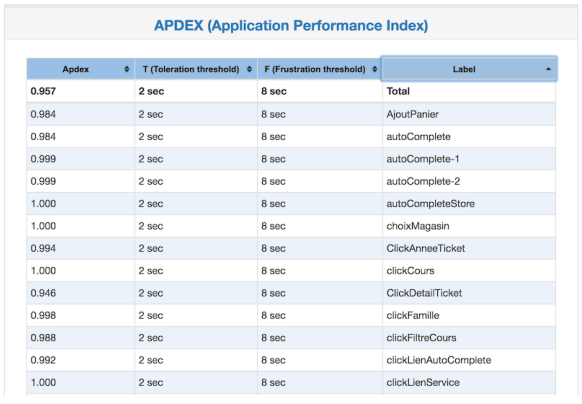


Descripción: La interfaz gráfica de JMeter con su árbol de elementos (Test Plan), paneles para controlar samplers y configuraciones. Muestra cómo se construyen y organizan pruebas de rendimiento.



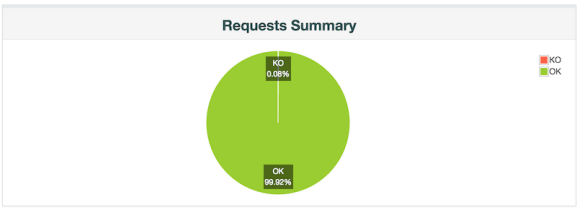
Test and Report informations

File:	"results-lt.csv"
Start Time:	"20/10 18:45:13"
End Time:	"20/10 20:46:04"
Filter for display:	"((^AjoutPanier)|(^Visite)|(^ClickAnneeTicket)|(^ClickDetailCommande)|(^ClickDetailTicket)|(^ClickMaCarteMaison)|(^ClickMesCoordonnees)|(^ClickMesTicketsCaisse)|(^ClickMonCompte)|(^ClickMonMagasin)|(^ClickMonSuiviCommande)|(^ClickOngletCommunaute)|(^ClickOngletConseils)|(^ClickPlusMoins)|(^CommunautePagination)|(^Continuer)|(^ContinuerMaCommande)|(^FicheProduit)|(^HP)|(^MajDonneesPerso)|(^MeConnecter)|(^PasserMaCommande)|(^PasserPro)|(^PositionnerModeLivraison)|(^PositionnerModeRelais)|(^PositionnerModeRetrait)|(^RechercheParCp)|(^RechercheParCritere)|(^RechercheParRef)|(^SaisieCP)|(^SelectionJour)|(^SelectionMagasin)|(^SelectionSecteurActivite)|(^ValiderAdresseLivPart)|(^ValiderAdresseLivPro)|(^VoirPanier)|(^autoComplete)|(^autoCompleteStore)|(^choixMagasin)|(^choixMagasinParCodePostal)|(^clickCours)|(^clickFamille)|(^clickFiltreCours)|(^clickHistoCommandeAnnee)|(^clickLienAutoComplete)|(^clickLienService)|(^clickMagasinsEtServices)|(^clickPagination)|(^clickRecherche)|(^clickSousUnivers)|(^clickUnivers)|(^cours-de-bricolage)|(^familleTriParUrl)|(^location-vehicules)|(^paginationFamille)|(^rechercheMagParGps)|(^sc1-recherche)|(^sc2-descente-produit)|(^sc3-commande-lad)|(^sc3-commande-retrait-mag)|(^sc3-commande-retrait-relais)|(^sc4-compte-internaute)|(^sc5-services)|(^selectionPointRelais)|(^showConcept)|(^triResultatsRecherche)|(-success|-failure)?"



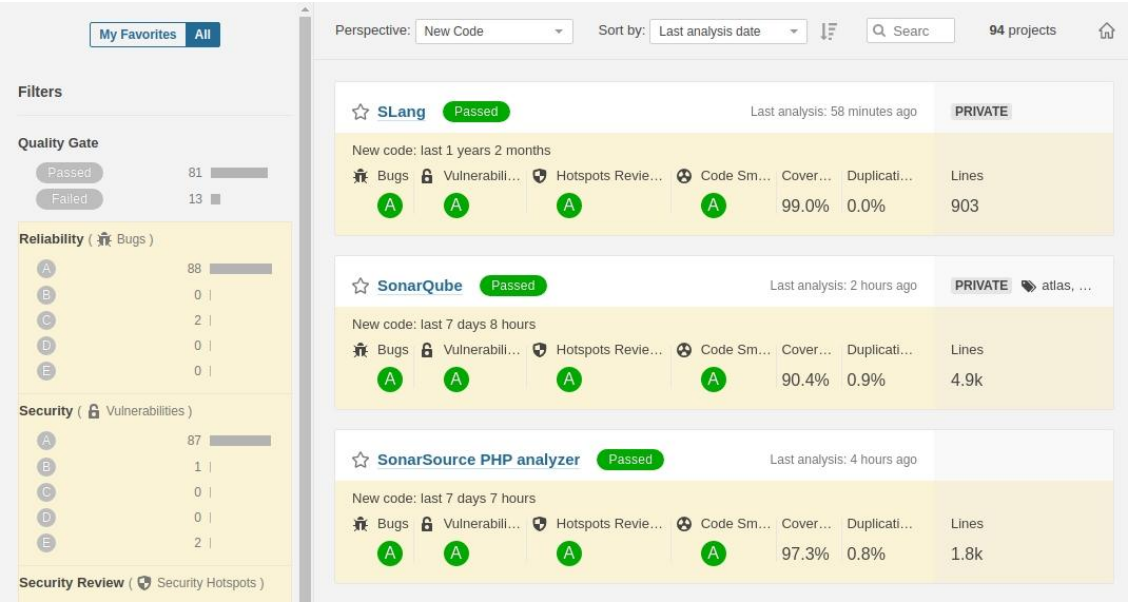
APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.957	**2 sec**	**8 sec**	**Total**
0.984	2 sec	8 sec	AjoutPanier
0.984	2 sec	8 sec	autoComplete
0.999	2 sec	8 sec	autoComplete-1
0.999	2 sec	8 sec	autoComplete-2
1.000	2 sec	8 sec	autoCompleteStore
1.000	2 sec	8 sec	choixMagasin
0.994	2 sec	8 sec	ClickAnneeTicket
1.000	2 sec	8 sec	clickCours
0.946	2 sec	8 sec	ClickDetailTicket
0.998	2 sec	8 sec	clickFamille
0.988	2 sec	8 sec	clickFiltreCours
0.992	2 sec	8 sec	clickLienAutoComplete
1.000	2 sec	8 sec	clickLienService

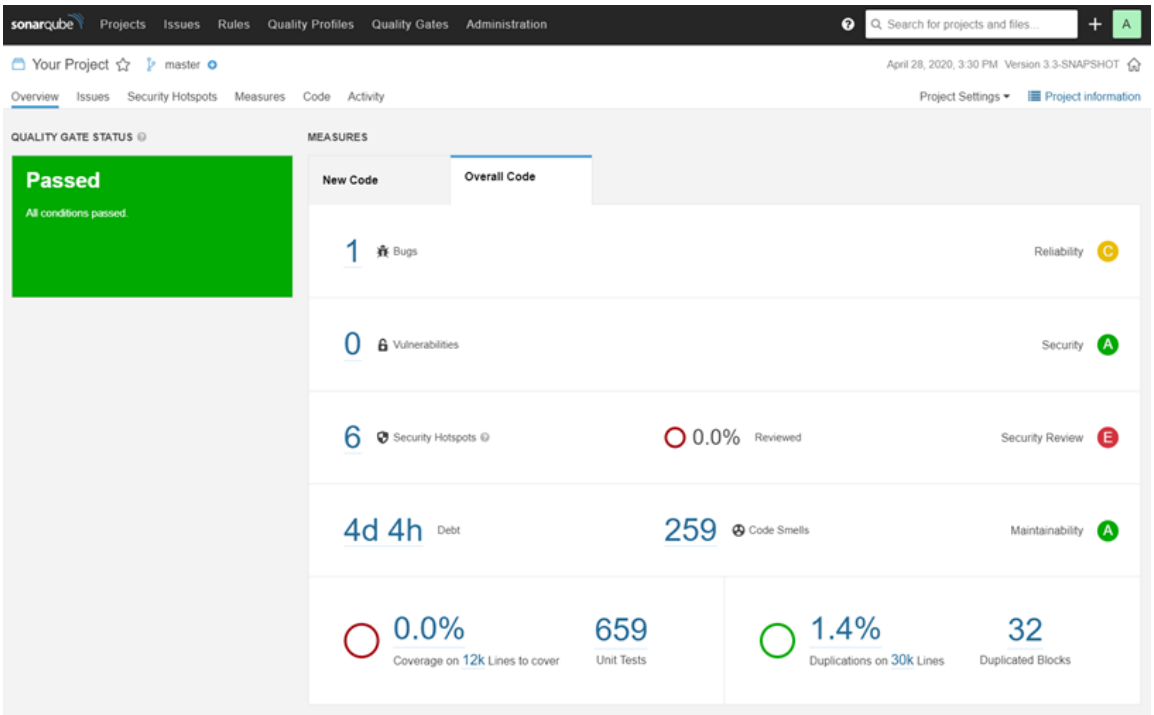


Descripción: Un ejemplo del reporte Dashboard generado por JMeter: gráficos de tiempos, errores, rendimiento y otros indicadores clave. Ideal para visualizar los resultados de las pruebas de carga. Captura #2

3- SonarQube



Descripción: Se muestra una sección del dashboard de SonarQube con indicadores visuales (semáforos verdes/rojos) sobre calidad del código nuevo, estado de Quality Gates, y la situación general del proyecto. Útil para monitorear calidad continuamente.



Descripción: Vista del panel principal ("Home Page") de SonarQube, con resumen de métricas clave como cobertura de pruebas, bugs, vulnerabilidades y deuda técnica. Ideal para tener una visión general rápida del proyecto.