UNIVERSIDAD PRIVADA DE TACNA
INGENIERIA DE SISTEMAS

TITULO:

Comparativa de Frameworks de Pruebas

CURSO:

CALIDAD Y PRUEBAS DE SOFTWARE

DOCENTE:

Ing. Patrick Cuadros Quiroga

Integrantes:

Sivirichi Falcón, Ricardo Alonso	(2018060905)
Chambilla Maquera, Araceli Noemi	(2018060897)
Arenas Paz Soldan, Miguel Jesus	(2017059282)
Cotrado Marino, Ana Luz	(2018060907)

Tacna - Perú
2020

Resumen

En este articulo de investigación expondremos sobre los frameworks de pruebas de apis asi como también se hablara sobre lo que es una api y como funciona también compararemos dos frameworks de pruebas de apis.

Palabras Clave: Frameworks de pruebas.

Abstract

In this research article we will discuss about the testing frameworks of apis as well as they will also talk about what it is an api and how it works we will also compare two apis testing frameworks.

Keywords: Test Framework.

---

1 Introducción

Para producir software con un nivel de calidad insuciente puede ser un pésimo negocio. Según Stadish Group en su informe de 2015(Chaos Report)[1], el 52Solo], el 52Solo el 29 %, llegan a el éxito, No solo es suficiente con que una aplicación funcione completamente , acorde a su propósito , sino que también debe satisfacer otros aspectos en términos de seguridad , rendimiento, accesibilidad ,etc. Por eso el objetivo de cualquier organización que se dedique a la industria del software debe producir software de calidad . La calidad debe ser una exigencia obligada y cada vez son mas las empresas que consideran el aseguramiento de la calidad. Una de las partes importantes que componen las tareas propias de un sistema de aseguramiento de la calidad que son las pruebas de software. Existen pruebas unitarias, de integración, funcionales, de rendimiento, etc. En la actualidad existen herramientas que permiten generar y ejecutar scripts, pequeños programas que permiten interactuar con una aplicación dada, simulando las acciones que podría realizar un usuario real, estas herramientas ayudan a incrementar la productividad como frameworks de pruebas para Apis.

2 Desarrollo

2.1 ¿Que es un API?

API (Application Programen Interface) Es una interfaz informática con la cual se puede intercambiar y comunicar datos entre dos sistemas de software separado. API incluye varias funciones/subrutinas que puede realizar otro sistema de software. La API define las solicitudes que se pueden realizar, como realizar las solicitudes, los formatos de datos que se pueden utilizar, etc. entre dos sistemas de software.

2.2 API Rest

2.3 ¿Que son las pruebas de API?

PRUEBA DE API es un tipo de prueba de software que valida las interfaces de programación de aplicaciones (API). El proposito de las pruebas de API es verificar la funcionalidad, confiabilidad, rendimiento y seguridad de las interfaces de programación. En API Testing, en lugar de utilizar entradas y salidas de usuario estándar (teclado), utiliza software para enviar llamadas a la API, obtener resultados y anotar la respuesta del sistema. Las pruebas de API son muy diferentes de las pruebas de GUI y no se concentran en la apariencia de una aplicacion. Se concentra principalmente en la capa de logica empresarial de la arquitectura de software.

2.4 Frameworks de Pruebas de APIs

2.4.1 Postman

Herramienta que se utiliza, sobre todo, para el testing de API REST, aunque también admite otras funcionalidades que se salen de lo que engloba el testing de este tipo de sistemas.

Gracias a esta herramienta, además de testear,consumir y depurar API REST, podremos monitorizarlas, escribir pruebas automatizadas para ellas,documentarlas, simularlas, etc.

Quizás sea una de las herramientas mas utilizadas para hacer testing exploratorio de este tipo de sistemas. Puede que no sea la mejor forma de escribir pruebas automatizada, pero sin duda es una de las

mas favorables para equipos con poca experiencia en programación, y sobre todo para hacer testing de todo tipo en general de API REST.

Características

- Crear Peticiones, te permite crear y enviar peticiones http a servicios REST mediante un interface gráfico. Estas peticiones pueden ser guardadas y reproducidas a posterior.
- Definir Colecciones, mediante Postman podemos agrupar las APIs en colecciones. En estas colecciones podemos definir el modelo de autentificacion de las APIs para que se añada en cada petición. De igual manera podemos ejecutar un conjunto de test, ası como definir variables para la colección.
- Gestionar la Documentación, genera documentación basada en las API y colecciones que hemos creado en la herramienta. Además esta documentación podemos hacerla publica.
- Entorno Colaborativo, permite compartir las API para un equipo entre varias personas. Para ello se apoya en una herramienta de colaborativa en Cloud.
- Genera código de invocación, dado un API es capaz de generar el código de invocación para diferentes lenguajes de programacion: C, cURL, C, Go,Java, JavaScript, NodeJS, ObjectiveC, PHP, Python, Ruby, Shell, Swift,etc.
- Establecer variables, con Postman podemos crear variables locales y globales que posteriormente utilicemos dentro de nuestras invocaciones o pruebas.
- Soporta Ciclo Vida API management, desde Postman podemos gestionar el ciclo de vida del API Management, desde la conceptualizacion del API, la definición del API, el desarrollo del API y la monitorización y mantenimiento del API.
- Crear mockups, mediante Postman podemos crear un servidor de mockups o sandbox para que se puedan testear nuestras API antes de que estas estén desarrolladas

Ventajas

- Permite la colaboración entre miembros del equipo.
- Tiene una interfaz mas intuitiva y atractiva.
- Posee extensión para Google Chrome, por lo tanto no es necesario instalar la aplicación de escritorio.
- Como ya mencionamos en la descripción, tiene una opción muy interesante, que son las coleccines, que funciona básicamente como una base de datos de peticiones.
- Es extensible y se puede integrar con otras herramientas, por ejemplo ejecutando las suites de prueba desde un motor de CI/CD.
- Permite agregar scripts en lenguaje Javascript para agregar validaciones, configurar y/o automatizar pruebas (esto se realiza directamente en la petición).

2.4.2 SoapUI

SoapUI es una herramienta desarrollada en Java, utilizada para pruebas de aplicaciones con arquitectura SOA o REST. Soporta múltiples protocolos como SOAP, REST, HTTP, JMS y JDBC.

La herramienta cuenta con una versión de código abierto y otra versión paga desarrollada por la compañía SmartBear. La herramienta SoapUI se creo inicialmente para probar los servicios SOAP. Luego, se extendió a los servicios web RESTful. Para un usuario nuevo puede requerir una mediana curva de aprendizaje ya que la misma en ocasiones no resulta tan intuitiva.

Características

La herramienta ofrece muchas características avanzadas para las pruebas API, incluyendo:

- Genera pruebas fácilmente usando arrastrar y soltar, apuntar y hacer clic.
- Potentes pruebas basadas en datos con datos de archivos y bases de datos.
- Los scripts se pueden reutilizar fácilmente.
- Servicios de mocks con mocking RESTful.
- Pruebas asíncronas.

Ventajas

- Tiene una mejor integración que Postman para trabajar con el protocolo SOAP (ya que inicialmente estaba pensada para eso).
- Es un proyecto mas maduro y tiene más tiempo en el mercado.
- Es una aplicación más orientada a testing y no simplemente a consumir una API, documentarla y publicarla. Permite estructurar las pruebas en test suites, test cases y test steps.
- La ejecución de pruebas se puede integrar con herramientas tales como: Maven, motores de CI/CD, etc.
- Permite agregar scripts en lenguaje Groovy.Esto permite agregar validaciones, configurar y/o automatizar pruebas.

3 Comparativa Postman y SoapUI

- Generación automatizada de aserciones: esta capacidad implica analizar API y generar aserciones automáticamente. Se espera que ahorre tiempo al generar aserciones manualmente para las API. Las dos herramientas proporcionan funciones básicas para generar scripts de aserción basados en algunas plantillas o reglas predefinidas.
- Informes de prueba: las dos herramientas proporcionan capacidades para reportar resultados de prueba de API. Postman genera informes en formatos JSON y HTML. Para SoapUI, la capacidad de producir informes de prueba detallados se encuentra en la edición comercial.
- Lenguajes de scripting: todas las herramientas admiten lenguajes basados en Java.
- Pruebas de UI web y aplicaciones móviles: en el desarrollo de una aplicación multiplataforma, el equipo realiza pruebas de aplicaciones móviles y UI web además de las API de pruebas. Por lo tanto, existe un déficit entre Postman y SoapUI, ya que no permiten a los desarrolladores y tester usar la misma herramienta, compartir y colaborar en los mismos casos de prueba.

Frameworks de pruebas APIs		
Caracteristicas	Postman	SoapUI
Pruebas de API REST	Sí 3	Sí
Pruebas API de SOAP	No	Sí
Generación automatizada de aserciones.	Sí	Sí
Soporte basado en datos	JSON CVS	Sí (*)
Compatibilidad con Cucumber BDD	No	No
Manejo del entorno	Sí	Sí (*)
Informes de prueba	JSON simples, formatos HTML	Sí (*)
Lenguajes de scripting	JavaScript	Groovy
Web UI testing	No	No
Pruebas de aplicaciones móviles	No	No
Análisis de ejecución de pruebas	No	No
Prueba de reutilización de script	Ninguno	Sí

4 Conclusión

Las pruebas en un proyecto de APIs son fundamentales, nos garantizan que podemos hacer cambios de versiones, actualizaciones, corrección de bugs o nuevas implementaciones garantizando que la especificación coincide con el método y que cualquier cliente que va a consumir nuestras APIs lo puede hacer sin ningún problema.

5 Recomendaciones

5.1 Planificación previa

Desde el inicio de la estimación de casos de prueba, es aconsejable asignar prioridades de ejecución: Alta, Media, Baja y palabras clave como Regresión, Sanidad, etcétera para los casos de prueba. De esta manera se puede hacer un filtrado rápido por prioridades y etiquetas, que permitan diferenciar rápidamente cuáles deberían ser los casos más importantes.

5.2 Uso de técnicas de diseño de casos de prueba

Entidades como International Software Testing Qualifications Boardy American Society for Quality mencionan varias técnicas para la creación de casos de prueba que permiten una mejora,de tal manera que al aplicarlas se obtienen menos casos de prueba y un mayor factor de cobertura o bien una base de creación empírica. Al tener un menor número de casos de prueba, se reducen las pruebas exhaustivas y tiempos de ejecución.

5.3 Clasificación de suites

La división de casos de prueba según los módulos de un sistema, facilitará la selección de casos al momento de construir un Plan de Pruebas para Regresión. En las carpetas o suites deberían evitarse nombres ambiguos o diferentes a lo definido en el requerimiento, o en los nombres que se muestran en el sistema. Si la herramienta de manejo de casos de prueba lo permite, lo ideal es organizar las suites mediante jerarquías según la misma aplicación para que la selección de casos de prueba por componente pueda ser natural. Por ejemplo, si el módulo transferencias" tiene como opcio-

nes "Transferencias locales", "Transferencias a terceros", "Transferencias internacionales", debería existir un folder padre llamado "TRANSFERENCIAS", con 3 carpetas hijas, de manera tal que la carpeta llamada "Transferencias" tenga como opciones "Transferencias locales", "Transferencias a terceros", "Transferencias internacionales". Este tipo de organización es útil para mantener los casos de prueba ordenados y agilizar la búsqueda de casos de prueba. Sin embargo, no se debe abusar de la jerarquía, pues llegar a tener más de 4 ó 5 niveles de profundidad se torna poco eficiente.

5.4 Programación de mantenimientos

Conforme nuestra aplicación crece o sufre cambios y mejoras, se hace más necesario dedicar tiempo a revisar si estos cambios afectan nuestros casos actuales, invalidándolos o requiriendo actualización. Si sabemos que el cambio los afectará, se deberían incluir tareas de actualización. Si estamos bajo un modelo de un sistema antiguo con casos que nunca han tenido mantenimiento, se recomienda incluir tareas periódicas para la revisión de todos los casos existentes.

5.5 Definición del tipo de Regresión a realizar

No todas las pruebas de regresión implican una ejecución del 100 En estos casos es necesario un análisis de dependencias para tener certeza de que los cambios por aplicar efectivamente no afectan otras partes

5.6 Automatización

Las pruebas de regresión suelen ser procesos largos y al incorporar pruebas de automatización se consiguen varias ventajas. Primeramente, la capacidad de ejecución aumenta, al tiempo que la duración de las pruebas se reduce. Los scripts pueden ejecutarse tantas veces como se requiera sin que esto implique desgaste en el equipo. Asimismo, se pueden incorporar diferentes ambientes en la prueba sin que esto aumente el tiempo de las mismas, pues pueden ejecutarse en paralelo.

Referencias

[1] Freeman, E., Robson, E., Bates, B., y Sierra, K. (2008). Head first design patterns. .ºˈ Reilly Media, Inc.".

[2] Dockins, K. (2017). Design Patterns in PHP and Laravel. Apress. https://allitbooks.net/web-development/2056-design-patterns-php-laravel.html

[3] FEDERICO TOLEDO,(2020),API testing con Postman y SoapUI. https://www.federico-toledo.com/api-testing-con-postman-y-soapui/

[4] SHANE HASTIE,S.H., STEPHANE WOJEW-ODA, S. W. ,(2015),Standish Group 2015 Chaos Report QA with Jennifer Lynch. InfoQ.https://www.infoq.com/articles/

[5] MICROSOFT.(S. F.).CODIGOS DE ERROR DE LA API REST,MicrosoftDocs. Recuperado el 2 de diciembre de 2020,de:https://docs. microsoft.com/es-es/partner/develop/