**Introducción**

Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP[[1]](#footnote-1) y URI[[2]](#footnote-2), describe en su tesis de doctorado (Fielding, 2000) a REST (Representational State Transfer) como un estilo arquitectónico para sistemas hypermedia distribuidos y muestra cómo estos estilos se pueden utilizar para guiar el diseño de nuevas arquitecturas de software basadas en la red. Según Fielding, REST provee un conjunto de restricciones arquitectónicas que, cuando son aplicadas en su totalidad enfatizan la escalabilidad de las interacciones de los componentes, la generalidad de las interfaces, el despliegue independiente de componentes y componentes intermedios, reducir la latencia de la interacción, reforzar la seguridad , y encapsular los sistemas más antiguos. Las restricciones especificadas por Fielding son las siguientes:

* Debe ser un sistema cliente-servidor.
* Debe ser sin estado, es decir, cada petición debe ser independiente de las otras.
* Tiene que soportar un sistema de caching[[3]](#footnote-3).
* Debe proveer una interface uniforme, todas las interacciones a través del sistema se deben llevar a cabo vía una interface standard y común.
* Debe estar diseñado en capas para soportar la escalabilidad.
* Debe proveer código bajo demanda, esto es, permitir extender la funcionalidad del cliente descargando y ejecutando código en la forma de applets o script. Esta es una restricción opcional.

El termino RESTful es típicamente utilizado para referirse a los servicios web que cumplen con las restricciones definidas en la arquitectura REST. Por esta razón, las aplicaciones web dinámicas no siempre son RESTful, porque pueden no cumplir con alguna restricción (Leonard Richardson, 2007)

En general, desarrollar servicios basados en REST es una tarea difícil. No sólo por el esfuerzo dedicado al diseño y la implementación, si no por el tiempo que debe dedicarse al testing en las etapas de desarrollo, a su mantenimiento en las etapas posteriores, y a la documentación para que los desarrolladores que vayan a usarla tengan claro su funcionamiento.

Es por ello que resulta de gran importancia contar con herramientas que faciliten la creación y ejecución de los casos de prueba para validar el correcto funcionamiento de los servicios provistos, así como también proveer la documentación actualizada para estos servicios.

**Motivación**

Existe gran cantidad de herramientas que permiten realizar pruebas sobre servicios web basados en REST de forma manual (algunos ejemplos, SoapUI (SoapUI), Curl (cURL), RestClient (RestClient tool)) pero realizar estas pruebas de forma manual representa un gran esfuerzo humano así como también tiempo invertido.

Por otro lado, las opciones de automatización disponibles consisten en la implementación de un framework de prueba combinando distintas herramientas o tecnologías (por ejemplo Apache HttpClient (HTTP Client), TestNG (TestNG), REST-assured (REST-assured), entre otras) lo que requiere el conocimiento de la herramienta seleccionada, el lenguaje de programación utilizado (por ejemplo Java) y de las metodologías de desarrollo de software requeridas para la implementación del fremework (por ejemplo programación orientada a objetos).

Además, es necesario contar con una completa y actualizada documentación de cada uno de los servicios disponibles para poder hacer uso de dicha API, así como también para poder realizar tanto las pruebas de forma manual como automática.

En mi actual trabajo donde desempeño tareas de gestión de calidad de Software estamos realizando el desarrollo de una API basada en REST, y nos hemos encontrado en la necesidad de automatizar tanto la generación de la documentación de dicha API como también las pruebas sobre la misma.

En base a todo lo enumerado anteriormente y luego de haber analizado las herramientas disponibles para atender estas necesidades he diseñado he implementado una herramienta que permite generar la documentación de una API REST de forma automática y utilizar esta misma documentación para automatizar las pruebas sobre la misma.

NOMBRAR VENTAJAS, esta herramienta permite …..

**Objetivos**

El Objetivo de este trabajo es desarrollar una herramienta que permita la automatizar tanto la generación de documentación de una API basada en REST así como también automatizar la ejecución de las pruebas sobre este tipo de APIs.

La creación de los casos de prueba se realizará a partir de la documentación generada dinámicamente desde la propia Api.

Gracias a la interfaz que esta herramienta posee, esta puede ser utilizada por desarrolladores y testers durante la etapa de desarrollo de la API, así como también por personas que no tengan conocimientos técnicos ya que el mecanismo de creación de los casos de prueba será a través de una interfaz gráfica que muestre la lista de servicios disponibles obtenidos automáticamente desde la documentación de la API.

La herramienta nos muestra la documentación de la API (servicios disponibles, métodos disponibles para cada servicio, parámetros requeridos, formato de la respuesta esperada, entre otros), a partir de esta lista de servicios podemos crear y ejecutar casos de prueba de forma automática.

La documentación se obtiene con la ayuda de Swagger (Swagger) el cual es un framework para documentar servicios REST a partir del código fuente de la Api. Para ello, hace uso de diversas anotaciones dentro del código de los servicios, para luego ofrecer mediante JSON o XML la documentación recogida a través de las anotaciones.

Esto permitirá que la documentación de los servicios se actualice al mismo tiempo que se actualiza el servidor. Figura 1.

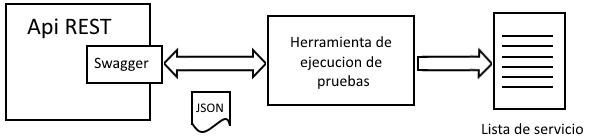


Figura 1: obtención de la lista de servicios disponibles en la API

La documentación que provee Swagger (Swagger) de dichos servicios está compuesta por los métodos a llamar, desde que URLs, cuáles deben ser los parámetros de entrada, cual serán los resultado esperado y el código de respuesta, entre otros.

Mediante la lista de servicios obtenida desde el framework de documentación, la herramienta es capaz de mostrar al usuario la lista de servicios disponibles en la Api. De esta forma, a través de la interface de usuario, es posible crear los casos de prueba para testear dichos servicios (Fiura 2).



Figura 2: creación de casos de prueba a partir de la lista de servicios

Esto facilita la creación de los casos de prueba ya que no es necesario que el usuario tenga presente cada uno de los servicios disponibles sino que los podrá seleccionar para su incorporación en algún caso de prueba.

La herramienta también permite crear conjuntos de casos de prueba. Cada caso de prueba se compone de pasos y cada paso está formado por un llamado a servicio y el resultado esperado.

Al realizar la ejecución de los casos de prueba, se realizan los llamados a los servicios correspondiente, estos llamados se realizan contra el servidor donde se encuentran alojados los servicios REST, para luego validar el resultado esperado contra el valor obtenido, y de esta forma determinar el resultado del test. Los resultados de la ejecución de cada caso se plasmados en un informe indicando cuales casos fallaron y cuales pasaron la prueba. Ver figura 3

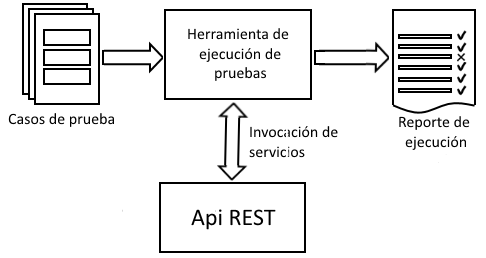


Figura 3: Ejecución de casos de prueba invocando los servicios de la API. Se obtienen como resultado los reportes de ejecución de las pruebas.

Para la implementación de los componentes que realizan las solicitudes contra el Api REST se utilizara Rest-Assured (REST-assured) el cual es un Java DSL (Domanin Specific Languaje) para facilitar la construcción de las pruebas a ejecutar.

1. Hypertext Transfer Protocol o HTTP es el protocolo usado en cada transacción de la World Wide Web. [↑](#footnote-ref-1)
2. Un identificador de recursos uniforme o URI (Uniform Resource Identifier) es una cadena de caracteres que identifica los recursos de una red de forma unívoca. [↑](#footnote-ref-2)
3. Sistema intermediario entre el cliente y el servidor que almacenan copias de las respuestas de un servidor de tal forma que subsiguientes peticiones pueden ser respondidas por el propio caché, permitiendo reducir el ancho de banda consumido, la carga de los servidores y el retardo en la descarga. [↑](#footnote-ref-3)