**Seccion 3: Webservices testing with SOAP UI & Ready API tools**

**8. Introduction to SOAPUI, Installing SOAPUI & Working with SOAPUI features:**

* Soap Ui es otra herramienta usada para probar SOAP services y luego se implementó para probar también REST
* En postman se puede hacer solo el testing funcional, solo se envia el request, se revisa la respuesta correcta y se pueden poner puntos de validación
* En SOAP UI se tiene muchas otras ventajas, se puete hacer por ejemplo testing de seguridad, de performance o load testing.
* Ventajas SOAP UI:

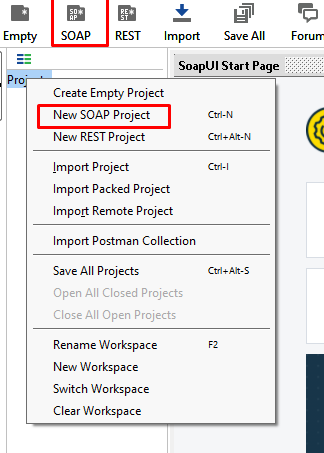
1. Funcional testing
2. Security Testing
3. Performance/load testing
4. Supports both SOAP & Rest services

* Soap viene en 2 formatos, SOAPUI Pro el cual es licenciado y SOAPU opensource. El testing de Seguridad, performance y load solo están habilitadas en el pro
* Lo primero que se debe hacer es descargar e instalar SOap ui, se busca en Google y se instala.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Para trabajar con SOAP se necesita el WSDL (Web service Description language) esto es un XMl.
* Primero se debe crear un proyecto, adentro de este se puede crear un test suite y luego dentro de este ultimo unos test cases, esta es la jerarquía
* Para revisar los WSDL se instala el Wizdler plugin para el navegador.
* Se utilizará el siguguiente WSDL para esta parte[www.dneonline.com/calculator.asmx?wsdl](http://www.dneonline.com/calculator.asmx?wsdl)
* Primero se debe crear el proyecto en soap. Para esto hay varios opciones:



* Luego de darle en crear se pone un nombre y se debe agregar el archivo. En este caso lo descargamos de la pagina anteriormente mencionada

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Si lo que se tiene es el url del WSDL también se puede pegar allí
* Las opciones que allí se muestran son muy importantes. Si se selecciona la primera crea un test case para todos, y se selecciona la segunda crea un test case para cada uno. En este caso vamos a seleccionar el primero

Imagen que contiene Gráfico

Descripción generada automáticamente

* Si se le da click a una de las request se abre una ventana con el body

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Ahora se crearán una test suite y test cases para poder probar las request
* Al darle click derecho sobre el proyecto se puede selccionar el “Generate test suite”

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* Esto nos lleva a otra pantalla

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* En esta pantalla también son importantes las opciones de crear un test case para cada operación o un solo test para todas las operaciones. Se presiona Ok
* Ahora bajo el proyecto aparece unos nuevos ítems

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Imagen que contiene Diagrama

Descripción generada automáticamente

* Clickeando en el test case se abre una ventana donde se pone el input en la ventana mas a al derecha sale la respuesta

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto, Pizarra

Descripción generada automáticamente

* Acá se han ejecutado todos los test cases individualmente
* En la parte de debajo de la ventana hay un item llamado “Assertions”, al clickearlo se abre y allí es donde se pondrán las verificaciones

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Se abre una ventana con todas las opciones que se tienen

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Le damos en contains que es uno de los mas usados.
* Se pone algo que debe estar en la respuesta

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

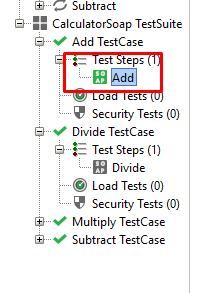
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* El status del caso de prueba cambia cuando el assertion se valida



* Los casos se pueden ejecutar secuencialmente o también en paralelo.
* Se le da doble click al test suite y esto nos abre otra ventana. En la parte superior de esta se pude seleccionar si se quieten correr de manera secuencial o de manera paralela

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Tabla

Descripción generada automáticamente con confianza mediaTabla

Descripción generada automáticamente

* Una parte importante es la documentación. Para tener la documentación de ese servicio, se le da click derecho y luego generate documentation

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* Esto pide indicar un folder, y luego de esto, genera una documentación

<file:///C:/Users/ASUS/Documents/Cursos/Learn%20API%20&%20Microservices%20Testing/Seccion%203/8/SOAP%20UI%20doc/wsdl-report.html>

Imagen que contiene Texto

Descripción generada automáticamente

* Agregamos un nuevo Proyecto:

<http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?wsdl>

* En este casio vamos a vrear un solo test case para todas las request. Por efectos prácticos se van a remover algunos de los ítems

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza baja

Texto

Descripción generada automáticamente

* Hasta el momento se ha visto como probar Soap services, ahora se mirará como hacerlo con Rest Services
* En Rest necesitamos un archivo WADL (Web appplication description language) (URL). El body puede ser XML, Jason o cualquier formato
* Usaremos : [www.Restcountries.eu](http://www.Restcountries.eu) . Opción gratis pero solo 50 request por mes. Se accede y se pide la key
* Para crear el proyecto en la parte superior se la de REST y se siguen los pasos

Texto

Descripción generada automáticamente

* Para poder recibir la respuesta se debe poner como parámetro la key asignada

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* Para crear una Test suite se hace igual a con SOAP.

**9. Assertions in SOAPUI:**

* Assertions es otro nombre para un punto de verificación. Son puntos de validación
* Assertions: verificaciones
* SOAP service 🡪 XML
* Rest servixec 🡪 Jason
* Se usará el siguiente WSDL: <https://ws.footballpool.dataaccess.eu/info.wso?WSDL>
* Se crea el proyecto, se pone la url y se siguen los pasos normales.
* Crearemos una test suite solo con el nombre de los jugadores

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Texto

Descripción generada automáticamente

* Se tiene diferentes tipos de assertions. Cada una de las mostradas tiene muchas otras adentro. En la pestaña scrips podemos escribir lo que queramos.

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* No solo se pueden verificar los datos, también los headers
* Agregamos el assertions ‘Contain’

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Imagen que contiene nombre de la empresa

Descripción generada automáticamente

* También se puede poner el contrario ‘not contains’

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

* En property content tenemos 2 assertions mas:

1. Xpath match: Se pone la direccion de un valor en especifico para obtener el valor por ejemplo:

Texto, Carta

Descripción generada automáticamente

Vamos a sacar el Id el primer jugador, como hay tantos jugadores con el mismo path se debe especificar el jugador:

Texto, Carta

Descripción generada automáticamente

El path sería: //m:AllPlayerNamesResponse/m:AllPlayerNamesResult/m:tPlayerName[1]/m:iId

Al seleccionar el assertion se debe clickear el botón the ‘Declare’, esto añade unos paths por defect o luego se copia nuestro path . en la segunda parte se debe poner el valor que esperamos. Si clickeamos en ‘Select from current’, se agrega automáticamente el esperado

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Si se quiere por ejemplo sacar el numero de nodos, se puede sacar con el mismo assertions.

Esto se haría con el path count (//m:sName)

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Verifiquenos si un nombre tiene solo letras, sin símbolos ni números:

matches(//m:AllPlayerNamesResponse/m:AllPlayerNamesResult/m:tPlayerName[1]/m:sName,'[A-Z,a-z]\*')

Los Id tiene solo números:

matches(//m:AllPlayerNamesResponse/m:AllPlayerNamesResult/m:tPlayerName[1]/m:iId,'[0-9]')

o

matches(//m:AllPlayerNamesResponse/m:AllPlayerNamesResult/m:tPlayerName[1]/m:iId,'\d[1-4]') -- Enm este se verifican los digitos

1. XQuery match: es diferente al Xpath pero no es tan usado, Este es con SQL language. Esta opción no es tan usada.

* Para verificar el Status code se pueden usar otros assertions ‘Valid status code’
* También existe el ‘Invalid status code’.
* El assertion ‘Soap fault’ verifica si lo que se recibe es SOAP o no
* En el assetions script se puede escribir lo que se quiera, En el documento hay varios ejemplos.
* Para los servicios REST se tienen los mismos assertions.
* Usaremos el siguiente URL: <https://reqres.in/api/users?page=2>
* Creamos la test suite como siempre
* Hay algunas assertions que solo se activan con JSON
* Para hayas los nodos en JSON se tiene el JSONpath fidner
* JSON path count: Cuantos nodos tenemos en el json
* Jsonpath match: Verifica el valor
* Jsonpath existence: No da los datos, solo true or false
* Jsonpath Regex match :Hace match con todo, si por ejemplo se le ingresa Tob.\* saldrá true
* Los scripts también se pueden usar con JSON, en el documento hay varios

**10. Properties in SOAPUI, Property Transfer & Data Driven Testing:**

* Propiedades en SOAPUI son variables que contienen un valor, este valor puede ser usado en cualquier lugar.
* Usaremos el WDSL <https://footballpool.dataaccess.eu/info.wso?WSDL>
* Creamos una test suite con all players with role
* En esta request se pasan 2 cosas, el nombre del país y del role



* En vez de pasar los nombres directamente , se puede crear una propidad para que acceda al valor y lo ponga donde se debe
* Vamos a crear una variable a nivel de la Test suite, para esto se le da click a la test suite y se abre una ventana en la cual se puede ver un item con “Properties”

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Para agregar una propiedad se abre la ventana y se pone el nombre



* Paa crear a nivel de proyecto también se le da doble click, luego en la ventana de properties y se pone lo que se necesite
* Lo mismo se puede hacer a nivel de Test
* Las properties pueden ser creadas en 4 niveles, proyecto, test suite, tese case y test step.
* Para crear en test step, se le debe dar click derecho en el test step. Luego add step, y selecciona properties. Se abre una venana nueva donde se pueden agegar

Texto

Descripción generada automáticamente

* Para accederlas se debe referencias en el request la propertie que se quiere usar, esto depende de la sintaxis

1. Project ${#Project#PropertyName}
2. TestSuite ${#TestSuite#PropertyName}
3. TestCase ${#TestCase#PropertyName}
4. TestStep ${TestStepName#PropertyName}
5. System ${#System#PropertyName}
6. Env ${#Env#PropertyName}
7. Global ${#Global#PropertyName}

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente



Texto

Descripción generada automáticamente

* Las properties también se pueden añadir mediante código de groovy
* En nivel de teststep se le da click al teststep añadir y groovy script

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Property transfer, o también llamado como value transfer. Es cuando se tiene una respuesta de una request y la respuesta se usa como un parámetro o input para otra request.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Para hacerlo se deben seguir varias reglas. Se debe tener crear un nuevo steop que se llama property transfer, se abre una nueva ventana. En esta se puede ver un source y un target. Se debe crear una variable de trasnferencia. Se seleccionan el source y tarjet, la propiedad que se desea capturar y a donde llevarla, y además el xpath.

Se da click en la opción namespace lo que trae ciertas variables

Se agrega entonces el path, en ete caso para el primer estadio: //m:StadiumNamesResponse/m:string[1]

También se puede ingresar como //\* m:string[1] el ¨indica todo lo que esta por encima de el stiring.

Se pone también el path para el otro //\*foot:sStadiumName

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

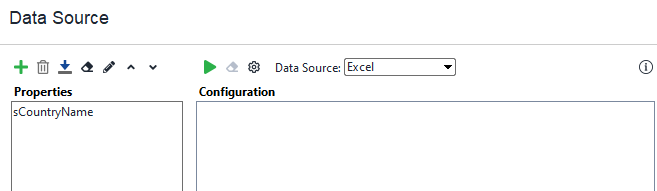
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Parametrization or Data Driven Testing (DDT)
* Es cuando pasamos cierta información por ejemplo desde un XML y corremos varias veces el request con la información que viene del XML. Se puede tomar la info de un XML, de una notepath o diferentes fuentes.
* Esta funcionalidad no está disponible en SOAPUi gratis, solo en la versión de pago(profesional). Esa versión la podemos descargar por unos días gratis y nos provee muchas funciones adicionales.
* Descargamos entonces la versión profesional, el nombre es Ready API.
* El DDT se puede hacer desde archivos de Excel, archivos de texto, generación de datos, JDBC, Json, XML…..
* Se descarga Ready Api, se pide la licencia con el usuario creado en la pagina.
* Si se tiene soap ui y ready api en el mismo sistema se guardaran las mismas request y proyectos
* Creamos un nuevo test case con los countries
* Vamos a crear unos datos de pruebas en Excel para pasarlos al soapui
* Se clickea en el test case, se agrega un nuevo step llamado DATA Source, se le da en aceptar en la ventana que sale y luego se genera un nuevo paso automáticamente el cual se llama Data source loop. Se abre una nueva ventana, en al cual selccionamos Excel



* El mismo numero de columnas que se tiene en el Excel deben ser el mismo numero de propiedades.
* Luego se selecciona el Excel, la hoja en la cual están los datos

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* Luego cambiamos el nombre de la propiedad

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Damos click en el botón de run y aparecen los valores en la parte de abajo

Imagen que contiene Texto

Descripción generada automáticamente

* Ahora se va a la request y se debe poner algo para llamar los valores del archivo. Se puede poner el statement en el request. O se puede ir a la pestaña de form y usar el browse

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Se abre una nueva ventana en la que se puede seleccionar el data source, el archivo

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

* Luegio se va al nivel del Test case y se corre

Tabla

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

* Ahora creamos un archivo de texto
* Se hace el mismo proceso pero se sube el TXT. En el loop se cambia de donde toma y el objetivo. Luego se deshabilita el XLS y por último se corre.
* Otro forma es la generación de datos directamente desde Ready Api, se crea el nuevo data source y se selecciona Data Generator. En esta opción hay varias propiedades por defecteo que se pueden usar

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* En el request se cambia el source y se modifica el loop
* Lo mismo se puede hacer con los otros tipos de datos.

**11. Data Driven Testing Using Multiple Data Sources in SoapUIPro (ReadyAPI)**

* Hay muchas fuentes de datos que se pueden usar: TXT, XML; Data base, etc.
* Se verá entonces como traer la data de varias fuentes y luego validar los resultados
* En los documentos de la clase se tiene unos archivos, primero usaremos el XLS, en este esta el nombre del país y el código correspondiente

Tabla

Descripción generada automáticamente

* Creamos un nuevo test suite, step y case. Y agregamos también una Data source
* Buscamos el XLS en los archivos y agregamos

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Corrremos para verificar los datos

Imagen que contiene Aplicación

Descripción generada automáticamente

* Vamos al request y agregamos el data source, en este caso es el valor countries de el archivo

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Se debe verificar entonces el country code cuando llegue la respuesta. Verificando en la respuesta vemos que el item que necesitamos tomar es:



Damos click derecho en este código y damos en add assertion y luego for content

Esto abre una nueva ventana en donde valida lo que se tiene

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Lugo le damos click en select content y seleccionamos de donde va a sacar la info

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Ahora verificará cada respuesta
* Se añada el loop para poder tomar todos los valores. Se configura y ya se puede correr.

Tabla

Descripción generada automáticamente

* Lo mismo se puede hacer con un text file. En este caso se deben crear las propiedades e indicar con que se separan en el archivo (Por alguna razón en este momento solo está permitiendo 2 y a la 3ra revienta)
* Otra data source es GRID. En esta se agregan las propiedades (Cada columna) y se pueden rellenar

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Otro tipo de source es el JDBC se usa con las data bases SQL, etc
* Para usar JDBC se debe:

1. Descargar y Copiar el driver de JDBC el cual se debe mantener dentro de la locación de ready api. Dentro de bin > ext
2. Tener un connection String con la data base: Hostname, port, sid/service, user password

* Se selecciona entonces el data source JDBL y se rellenan los campos
* Se conecta y se selecciona la data base que se quiere usar
* La ejecución de casos de prueba y generación de reportes se pueden hacer de 2 maneras, desde Ready aPi, test runner y desde el command promt.
* Los casos se pueden ejecutar secuencialmente o paralelamente
* También se puede sacar reporte

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

<file:///C:/Users/ASUS/Documents/Cursos/Learn%20API%20&%20Microservices%20Testing/Seccion%203/11/index.html>

* Para correrlo en test runner se da click en el test suit y correr con test runner. Se abrirá una nueva ventana. Si se da en get command line, se abre una nueva ventana con el comando para correrlo en el command promt
* Si se la da lunch abre al consola del programa y empieza a ejecutar

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Para correr en el commant promt se necesita exportar la suite primero (XML). Click derecho en la suite y exportar
* Se debe ir a la carpeta de Smart bear, abrir allí el cmd , agregarle el nombre del archivo y empieza a correr. También se puede copiar desde el test run

**12. Testing Flicker & Google maps Web service API’s, SwaggerIO:**

* Flickr y google maps son opensource, lo unico es que se debe generar la key necesaria para utilizar las Apis. Estas son REST APIs
* En los archivos de la clase se pueden ver los tutoriales para descárgalo
* Se descarga el proyecto desde <https://www.soapui.org/resources/tutorials/flickr.html>
* Se crea la key en <https://www.flickr.com/services/apps/create/>

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* Impotamos el archivo XMl en soap ui
* Se crea una test suite y se agrega la s en el http para que cumpla las condiciones
* Se va al nivel del proyecto y se abre para agregar la keya este nivel

Imagen que contiene Gráfico

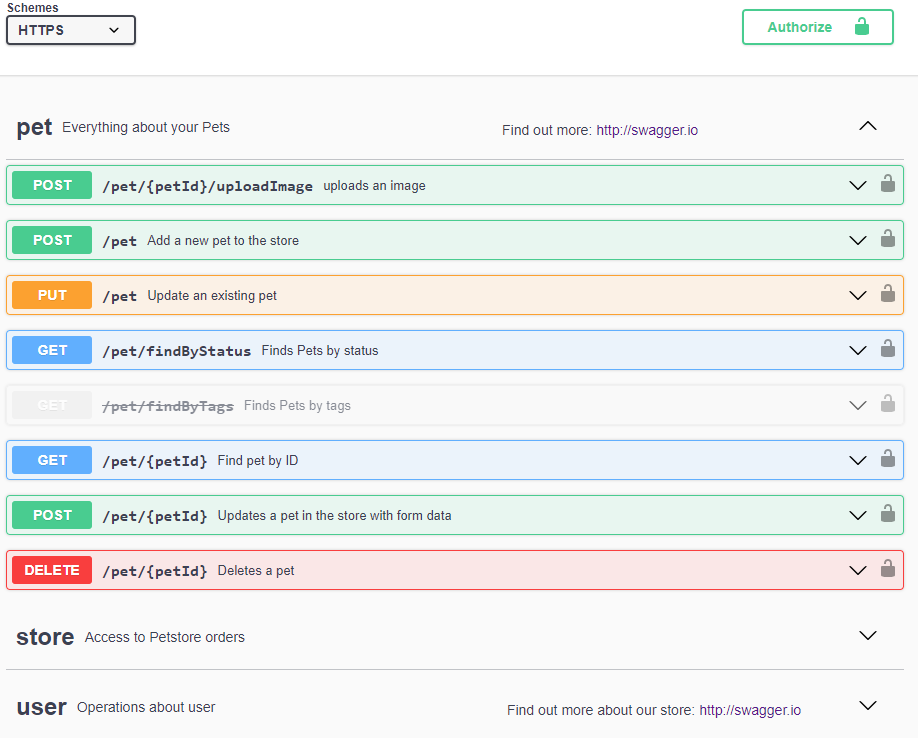
Descripción generada automáticamente

* En la documentación de soap ui también se puede buscar el proyecto de Google maps

Tabla

Descripción generada automáticamente con confianza media

* Haya algunas de las APIS que no necesitan API key, otras que si. Se va al link <https://developers.google.com/maps/documentation/javascript/get-api-key>
* Se debe crear un nuevo proyecto , se va acredenciales y se crea una nueva clave de API.
* Para algunas se debe habilitar el pago. En este caso no lo activaremos.
* **Swagger** juega un papel importante para la documentación. Es una herramienta para exponer la documentación de las APIs
* Swagger genera una Api interactiva para que los usuarios aprendan rápidamente sobre la API.
* Requirement – Design- Dev – Unit Testing luego de esto se mantiene la documentación de la API
* En Swagger IO se pueden tener interacciones con el server.
* Swagger no es para el testing, alli no se pueden tener test cases, no se pueden tener historias
* Un ejemplo se puede ver en <https://petstore.swagger.io/>



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Se le puede dar en execute y se ven la respuesta, esto no es para testear, es para ver como funciona
* Swagger esta hecho para REST no para SOAP
* Estas Request, puede ser llevadas a psotman y hacer el llamado normalmente, se debe lleva el URL, el body, etc…
* En el archivo de la clase hay documentación otros swagger para probar