TEST QA FALABELLA FINANCIERO

El siguiente test medirá los conocimientos teóricos y prácticos del participante para el cargo de Analista QA Automation en la gerencia de integración. Consiste en primer lugar de evaluar un caso hipotético de una API rest y generar un set de pruebas que asegure el correcto funcionamiento del servicio. En segundo lugar, se plantea realizar pruebas a una api publica y en tercer lugar se plantea una serie de conceptos que podrá describir con sus propias palabras, por ultimo una pequeña validación de sus conocimientos en js.

Favor enviar el test en un repositorio Github para validarlo

1.- Dado el siguiente escenario:

*Mockear ambiente y endpoint para esta prueba.*

*Dentro de los servicios digitales de banco falabella, se encuentra la revisión del estado de cuenta del cliente, los productos que tiene asignado, puntos acumulados, etc. De estos servicios se necesita realizar pruebas de contrato a la API que conecta el sistema de información de datos con la interfaz que maneja el cliente. La API arroja el siguiente archivo JSON:*

*{*

*Dato\_cliente: {*

*Nombre: string,*

*Apellido: string,*

*Direccion: string,*

*Email: string,*

*Telefono: number,*

*RUT: number*

*},*

*Dato\_perfil: {*

*Cuenta\_ppal: number,*

*Tipo\_cliente: string,*

*Tarjeta\_debito: number*

*},*

*Dato\_Sucursal: {*

*Sucursal\_nro: number,*

*Ejecutivo: string,*

*Telefono\_ejec: number,*

*Direccion\_suc: string*

*},*

*Dato\_productos: {*

*Tiene\_prod: boolean,*

*Tipo\_producto: [*

*Consumo,*

*Hipotecario,*

*Linea\_emergencia,*

*Linea\_sobregiro,*

*],*

*Puntos: number,*

*Tarjeta\_credito: number*

*},*

*Datos\_seguros: {*

*Tiene\_prod: boolean,*

*SOAP: string,*

*Seguro\_vehiculo: string,*

*Seguro\_fraude: string,*

*Seguro\_hogar: string,*

*Seguro\_vida: string*

*}*

*Datos\_fondos: {*

*Fondo\_mutual: string,*

*Ahorro\_previsional: number,*

*ctaRentaDiaria: string*

*}*

*Datos\_penalizaciones: {*

*InteresesMora\_Sobregiro: number,*

*InteresMora\_TarjetaCredito: number,*

*InteresMora\_Consumo: number,*

*InteresMora\_Hipotecario: number*

*}*

*}*

Realizar las siguientes validaciones y agregar las que considere necesaria:

* Validar la estructura del Json.
* Validar respuesta 200.
* Validar que los tipos de datos sean correctos.
* Validar Mensaje OK.
* Validar que el formato sea JSON.
* Validar que RUT sea un número.

pm.test("Pruebas", function () {   
pm.response.to.be.ok; // Valida mensaje OK

pm.response.to.be.json; // Valida formato JSON

pm.response.to.have.status(200); // Valida respuesta 200

pm.expect(tv4.validate(json, schema)).to.be.true; // Valida estructura JSON

});

2.- Desde la siguiente URL <https://restcountries.eu/#api-endpoints-name>, tomar el endpoint <https://restcountries.eu/rest/v2/name/>{name} y realizar las validaciones que se consideren necesarias, se deben escribir minimo 6 tests.

<https://restcountries.eu/rest/v2/name/>chile

* pm.test("Valida que exista Chile, function () {

pm.expect(pm.response.text()).to.include("Chile");

});

* pm.test("Valida que Content-Type esté presente", function () {

pm.response.to.have.header("Content-Type");

});

* pm.test("Valida que tiempo de respuesta sea inferior a 1000ms", function () {

pm.expect(pm.response.responseTime).to.be.below(1000);

});

* pm.test("Valida que status es 200", function () {

pm.response.to.have.status(200);

});

3.- Esta sección es para validar sus conocimientos sobre javascript:

function Sumar(a,b) {

return a + b;

}

function Producto(a,b) {

var c = 20;

return a\*b\*c;

}

Obtener en consola los resultados de las funciones de arriba.

* var sumaResult = Sumar(a, b);

console.log(sumaResult);

* var productoResult = Producto(a, b);

console.log(productoResult);

var Persona = {

nombre: ‘Juan’,

apellido: ‘Perez’,

telefono: 900199703,

email: ‘juan.perez@gmail.com’

}

Realizar una función que obtenga las propiedades del Objeto Persona.

pm.test ("Validar propiedades del objeto Persona", function () {  
var jsonData = pm.response.json (),  
propertiesToCheck = ['nombre', 'apellido', 'telefono', 'email'],  
body = JSON.parse (pm.environment.get ("body"));

propertiesToCheck.forEach((p) =>

pm.expect(jsonData).to.have.deep.property(p, body[p]));

});

4.- Dado el siguiente Wsdl http://216.10.245.166:8080/axis2/services/EmployeeManagementService?wsdl

4.1 Agregar TestSuite con las validaciones (testcases) que se consideren necesarias, agregar los assertions que se requieran según el request y el response obtenido, los assertions deben hacerse en groovy scripting, mínimo 6 assertions.

* pm.test('codigo es 200', function(){

pm.response.to.have.status(200);

})

* pm.test('Valida respuesta OK', function(){

pm.response.to.be.ok;

})

* pm.test("Valida que Content-Type header este presente", function () {

pm.response.to.have.header("Content-Type");

});

* pm.test("El cuerpo de la respuesta es un XML valido", function () {

pm.expect(xml2Json(responseBody)).to.exist;

});

4.2.- Construya un script que invoque a una clase groovy que devuelva el status code del servicio

def groovyUtils = new com.eviware.soapui.support.GroovyUtils( context )

def httpResponseHeaders = context.testCase.testSteps["testName"].testRequest.response.responseHeaders

def httpStatus = httpResponseHeaders["#status#"]

def httpStatusCode = (httpStatus =~ "[1-5]\\d\\d")[0]

log.info("HTTP status code: " + httpStatusCode)

Cualquier duda comunicarse por correo afrancoh@falabella.cl

Favor enviar el test en un repositorio Github para validarlo

**Buena Suerte!!!**