FORMATO .JSON

JSON y su utilidad.

JSON (acrónimo de JavaScript Object Notation, «notación de objeto de JavaScript») es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera (año 2019) un formato independiente del lenguaje.

Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que resulta mucho más sencillo escribir un analizador sintáctico (parser) para él. En JavaScript, un texto JSON se puede analizar fácilmente usando la función eval(), algo que (debido a la ubicuidad de JavaScript en casi cualquier navegador web) ha sido fundamental para que haya sido aceptado por parte de la comunidad de desarrolladores AJAX.

En la práctica, los argumentos a favor de la facilidad de desarrollo de analizadores o de sus rendimientos son poco relevantes, debido a las cuestiones de seguridad que plantea el uso de eval() y el auge del procesamiento nativo de XML incorporado en los navegadores modernos. Por esa razón, JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia (de aquí su uso por Yahoo!, Google, Mozilla, etc, que atienden a millones de usuarios) cuando la fuente de datos es explícitamente de fiar y donde no es importante el hecho de no disponer de procesamiento XSLT para manipular los datos en el cliente.

Si bien se tiende a considerar JSON como una alternativa a XML, lo cierto es que no es infrecuente el uso de JSON y XML en la misma aplicación; así, una aplicación de cliente que integra datos de Google Maps con datos meteorológicos en SOAP necesita hacer uso de ambos formatos.

Los tipos de datos disponibles con JSON son:

- Números: Se permiten números negativos y opcionalmente pueden contener parte fraccional separada por puntos. Ejemplo: 123.456
- Cadenas: Representan secuencias de cero o más caracteres. Se ponen entre doble comilla y se permiten cadenas de escape. Ejemplo: "Hola"
- Booleanos: Representan valores booleanos y pueden tener dos valores: true y false
- null: Representan el valor nulo.
- Array: Representa una lista ordenada de cero o más valores los cuales pueden ser de cualquier tipo. Los valores se separan por comas y el vector se mete entre corchetes. Ejemplo ["juan","pedro","jacinto"]
- Objetos: Son colecciones no ordenadas de pares de la forma <nombre>:<valor> separados por comas y puestas entre llaves. El nombre tiene que ser una cadena y entre ellas. El valor puede ser de cualquier tipo. Ejemplo:
  {"departamento":8,"nombredepto":"Ventas","director": "juan rodriguez","empleados":
  [{"nombre":"Pedro","apellido":"Fernandez"},
  {"nombre":"Jacinto","apellido":"Benavente"}]}

Al ser JSON un formato muy extendido para el intercambio de datos, se han desarrollado API para distintos lenguajes (por ejemplo ActionScript, C, C++, C#, ColdFusion, Common Lisp, Delphi, E, Eiffel, Java, JavaScript, ML, Objective-C, Objective CAML, Perl, PHP, Python, Rebol, Ruby, Lua y Visual FoxPro) que permiten analizar sintácticamente, generar, transformar y procesar este tipo de dato.

Hay muchos analizadores JSON en el lado del servidor, existiendo al menos un analizador para la mayoría de los entornos. En algunos lenguajes, como Java o PHP, hay diferentes implementaciones donde escoger. En JavaScript, el análisis es posible de manera nativa con la función JSON.parse(). Ambos formatos carecen de un mecanismo para representar grandes objetos binarios.

Con independencia de la comparación con XML, JSON puede ser muy compacto y eficiente si se usa de manera efectiva. Por ejemplo, la aplicación DHTML de búsqueda en «BarracudaDrive» (en inglés). Archivado desde el original el 21 de mayo de 2006. recibe los listados de directorio como JSON desde el servidor. Esta aplicación de búsqueda está permanentemente consultando al servidor por nuevos directorios, y es notablemente rápida, incluso sobre una conexión lenta.

Los entornos en el servidor normalmente requieren que se incorpore una función u objeto analizador de JSON. Algunos programadores, especialmente los familiarizados con el lenguaje C, encuentran JSON más natural que XML, pero otros desarrolladores encuentran su escueta notación algo confusa, especialmente cuando se trata de datos fuertemente jerarquizados o anidados muy profundamente.

Hay más comparaciones entre JSON y XML en JSON.org6

YAML es un superconjunto de JSON que trata de superar algunas de las limitaciones de éste. Aunque es significativamente más complejo,7 aún puede considerarse como ligero. El lenguaje de programación Ruby utiliza YAML como el formato de serialización por defecto. Así pues, es posible manejar JSON con bastante sencillez.

XML y sus aplicaciones para negocios.

XML o Extensible Markup Language, es un lenguaje de Tags o etiquetas que permite definir de un modo muy sencillo la estructura jerárquica a la que pertenece un dato, así como HTML permite definir la forma en que se muestra un dato en nuestro navegador.

XML permite, de un modo sumamente sencillo, estructurar la información de modo de enviarla con total seguridad de que el receptor sabrá que ese dato tiene una relación con otro dato dentro de la misma estructura enviada, también puede saber qué tipo de dato es el que está recibiendo (XML Schema), puede establecer cómo mostrarlo (XSL) e incluso cómo tiene que devolverlo (SOAP)

Así XML permite la comunicación de una aplicación a otra, o recibir y enviar datos estructurados mediante Internet sin tener que idear mecanismos complejos o excesivamente pesados para rearmar la información como en su origen.

XML tiene múltiples utilizaciones. La transmisión de datos es su origen, pero integrada con XML Schema se puede definir el tipo de dato que está viajando, o si se permiten valores nulos, repetidos, decimales o si se trata de un dato que mantiene una integridad referencial con otra información en el mismo documento transmitido.

XML es la fuente de SOAP, un protocolo basado en el estándar que permite el envío de paquetes de información bidireccional para la integración de aplicaciones remotas. Pudiendo de este modo transmitir datos por referencia e incluso en una transacción.

XML con XSL permite modelar la información visualmente para su presentación de modo de generar presentaciones dinámicas principalmente orientadas a B2C.

XML es un modo de parametrizar aplicaciones de forma sencilla, legible y comprensible tanto por aplicaciones como por personas y fácilmente accesible desde cualquier tipo de aplicación.

El 99% de las aplicaciones de escritorio actuales soporta lectura, escritura, importaciones y exportaciones a este formato para persistir la información de manera consistente, y con cada nueva versión XML se integra más en el Back Office de los sistemas de escritorio, gestión, Web, etc.

Todo esto con un modelo descriptivo en formato de texto, y basado en estándares de la industria definidos por el W3C (World Wide Web Consortium), que garantiza que la información podrá ser transmitida por Internet sin ningún tipo de traba (Firewalls) y que la interpretación de la misma es universal más allá de plataformas o lenguajes de desarrollo.

Una solución hoy día, no debiera cerrar la posibilidad de integración o comunicación con nuevas aplicaciones, módulos, funcionalidad o dispositivos. Tener en cuenta la transmisión e integración de información utilizando XML es un requisito necesario a la hora de establecer los alcances de una solución.

Por todo esto una solución tecnológica no es tal si no se ha analizado convenientemente la utilización de la infraestructura XML en la misma.

Sistemas de información Heterogéneos

Un ejemplo clásico y muy importante es el uso de XML en la integración de sistemas de información heterogéneos.

En la actualidad el mercado está inundado de aplicaciones específicas y/o verticales que junto a la existencia de aplicaciones generales y/o horizontales, lleva a que muchas veces se deban integrar aplicaciones desarrolladas sobre plataformas, modelos de datos y lenguajes distintos.

Dispositivos móviles

En la actualidad la movilidad del personal de una empresa es en muchos casos vital para su funcionamiento. La principal complicación en estos escenarios consiste normalmente en dotar al usuario del dispositivo móvil de información Inmediata, Oportuna y Actualizada proveniente del centro de datos.

Además, el usuario debe tener la posibilidad de modificar dicha información y actualizarla en el centro de datos sin tener que trasladarse físicamente, conectarse a la red y actualizar.

La tecnología móvil nos permite actualmente utilizar PDAs, Portátiles, teléfonos móviles, etc. que se puede comunicar con un servidor intercambiando información en con XML, WML y Servicios Web, y así optimizar la dinámica de la empresa contando con información fiable y actualizada en todo momento.

Servicios Web (Web Services)

Quizás la tecnología que más dará que hablar en el futuro inmediato es la relacionada con los Servicios Web. Esta nueva forma de transmisión de datos permite la comunicación bidireccional, con lo cual se pueden establecer comunicaciones entre aplicaciones utilizando protocolos estándares basados en XML como SOAP (Simple Object Access Protocol), y especificaciones (aunque no

estándares aún) como UDDI (Universal Description, Discovery and Integration) y WSDL (Web Service Definition Language).

Estas tecnologías encapsulan el XML en paquetes de transmisión o mensajes (SOAP), permiten la ubicación en internet de los Servicios Web existentes cual si se tratase de unas Páginas Amarillas de Web Services (UDDI), y dan la posibilidad de que la aplicación que hace uso del Servicio Web comprenda las interfaces de comunicación de este último (WSDL).

Este conjunto de definiciones permiten que las aplicaciones distribuidas se basen en tecnología abierta basada en estándares a diferencia de protocolos propietarios como DCOM o CORBA.

El resultado es la posibilidad de que aplicaciones de escritorio o Web se comuniquen con otras aplicaciones remotas para obtener datos o gestionarlos, como si se tratase de una aplicación local, sin importar la plataforma en que se encuentra cada una en tanto se respeten los estándares citados.

Así, si nuestra aplicación requiriese de la cotización actual de determinada Empresa, se puede buscar mediante UDDI un Servicio Web que brinde dicha información, utilizar las interfaces programáticas del mismo con WSDL, y finalmente comunicar nuestra aplicación con el Web Service mediante SOAP, sin necesidad de tener que preocuparse por cortafuegos (firewalls) que pudieran interrumpir la comunicación por tratarse de un estándar basado en texto plano y que se comunica por un protocolo montado sobre HTTP.

EJERCICIO 1

Transforma los datos en formato JSON mostrados a continuación en datos en formato XML y responde:

```
{
	"personas": [
		{
			"nombre": "Juan Pérez Galones",
			"altura": 1.72,
			"peso": 75,
			"pasatiempos": [
				"Comics",
				"Baloncesto",
				"Gaming"
				],
			"soltero": true,
			"direccion": {
				"calle": "Avenida Soleares",
				"numero": "44",
				"pais": "Chile"
				}
			},
		{
			"nombre": "Pedro Motos Sandez",
			"altura": 1.69,
			"peso": 72,
			"pasatiempos": [
				"Programación",
				"Senderismo"
```

```
                        ],
                    "soltero": true,
                    "direccion": {
                        "calle": "Torero Pedro Cano",
                        "numero": "78",
                        "pais": "México"
                        }
                    }
                ]
}
```


a) Muestra el código XML equivalente.


```
<?xml version="1.0" encoding="UTF-8"?>
<personas>
  <persona>
    <nombre>Juan Pérez Galones</nombre>
    <altura>1.72</altura>
    <peso>75</peso>
    <pasatiempos>
      <pasatiempo>Comics</pasatiempo>
      <pasatiempo>Baloncesto</pasatiempo>
      <pasatiempo>Gaming</pasatiempo>
    </pasatiempos>
    <soltero>true</soltero>
    <direccion>
      <calle>Avenida Soleares</calle>
      <numero>44</numero>
      <pais>Chile</pais>
    </direccion>
  </persona>
  <persona>
    <nombre>Pedro Motos Sandez</nombre>
    <altura>1.69</altura>
    <peso>72</peso>
    <pasatiempos>
      <pasatiempo>Programación</pasatiempo>
      <pasatiempo>Senderismo</pasatiempo>
    </pasatiempos>
    <soltero>true</soltero>
    <direccion>
      <calle>Torero Pedro Cano</calle>
      <numero>78</numero>
      <pais>México</pais>
    </direccion>
  </persona>
</personas>
```


b) Compara el número de caracteres que forma una codificación y otra. ¿Cuántos caracteres ocupa la codificación JSON? ¿Cuántos caracteres ocupa la codificación XML? (Nota: el número de caracteres se puede contar con un editor de texto).

XML: 830 caracteres
JSON: 517 caracteres

c) Transforma la notación JSON para que toda la información quede en una sola línea. ¿Crees que el contenido en una sola línea es equivalente al contenido inicial? ¿Qué ventajas e inconvenientes le ves a tener toda la información en una sola línea?

{"personas": [{"nombre": "Juan Pérez Galones","altura": 1.72,"peso": 75,"pasatiempos": ["Comics","Baloncesto","Gaming"],"soltero": true,"direccion": {"calle": "Avenida Soleares","numero": "44","pais": "Chile"}},{"nombre": "Pedro Motos Sandez","altura": 1.69,"peso": 72,"pasatiempos": ["Programación","Senderismo"],"soltero": true,"direccion": {"calle": "Torero Pedro Cano","numero": "78","pais": "México"}}]}

Tiene algunos caracteres mas pero la informacion se ve mucho mas desordenada y es mucho mas dificil ver su estructura a simple vista.

d) Transforma la notación XML para que toda la información quede en una sola línea. ¿Qué línea resulta más larga, la línea con notación JSON o la línea con notación XML?

<?xml version="1.0" encoding="UTF-8"?><personas><persona><nombre>Juan Pérez Galones</nombre><altura>1.72</altura><peso>75</peso><pasatiempos><pasatiempo>Comics</pasatiempo><pasatiempo>Baloncesto</pasatiempo><pasatiempo>Gaming</pasatiempo></pasatiempos><soltero>true</soltero><direccion><calle>Avenida Soleares</calle><numero>44</numero><pais>Chile</pais></direccion></persona><persona><nombre>Pedro Motos Sandez</nombre><altura>1.69</altura><peso>72</peso><pasatiempos><pasatiempo>Programación</pasatiempo><pasatiempo>Senderismo</pasatiempo></pasatiempos><soltero>true</soltero><direccion><calle>Torero Pedro Cano</calle><numero>78</numero><pais>México</pais></direccion></persona></personas>

La linea XML es mas larga debido a que hay un mayor uso de etiquetas que en JSON.

EJERCICIO 2

Copia el código de persona del comienzo y prueba a minificarlo en la web anterior.
Copia el resultado.
Localiza otra web que también permita minificar contenido.

https://codebeautify.org/jsonminifier

{"personas":[{"nombre":"Juan Pérez Galones","altura":1.72,"peso":75,"pasatiempos":["Comics","Baloncesto","Gaming"],"soltero":true,"direccion":{"calle":"Avenida Soleares","numero":"44","pais":"Chile"}},{"nombre":"Pedro Motos Sandez","altura":1.69,"peso":72,"pasatiempos":["Programación","Senderismo"],"soltero":true,"direccion":{"calle":"Torero Pedro Cano","numero":"78","pais":"México"}}]}

EJERCICIO 3

Analiza el siguiente código y responde a las preguntas indicadas más abajo:

{

```
	"marcadores": [
		{
			"latitude": 40.416875,
			"longitude": -3.703308,
			"city": "Madrid",
			"description": "Puerta delSol"
		},
		{
			"latitude": 40.417438,
			"longitude": -3.693363,
			"city": "Madrid",
			"description": "Paseo del Prado"
		},
		{
			"latitude": 40.407015,
			"longitude": -3.691163,
			"city": "Madrid",
			"description": "Estación de Atocha"
		}
	]
}
```


a) Obtén el código XML equivalente usando 2 webs de conversión on-line. Indica qué webs has utilizado, el resultado obtenido y si el resultado es el mismo con ambas herramientas. Si no es elmismo indica por qué crees que no es el mismo.

https://codebeautify.org/jsontoxml#


```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
	<marcadores>
		<latitude>40.416875</latitude>
		<longitude>-3.703308</longitude>
		<city>Madrid</city>
		<description>Puerta delSol</description>
	</marcadores>
	<marcadores>
		<latitude>40.417438</latitude>
		<longitude>-3.693363</longitude>
		<city>Madrid</city>
		<description>Paseo del Prado</description>
	</marcadores>
	<marcadores>
		<latitude>40.407015</latitude>
		<longitude>-3.691163</longitude>
		<city>Madrid</city>
		<description>Estación de Atocha</description>
	</marcadores>
</root>
```


https://www.freeformatter.com/json-to-xml-converter.html#ad-output

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <marcadores>
    <element>
      <city>Madrid</city>
      <description>Puerta delSol</description>
      <latitude>40.416874</latitude>
      <longitude>-3.703308</longitude>
    </element>
    <element>
      <city>Madrid</city>
      <description>Paseo del Prado</description>
      <latitude>40.41744</latitude>
      <longitude>-3.693363</longitude>
    </element>
    <element>
      <city>Madrid</city>
      <description>Estación de Atocha</description>
      <latitude>40.407017</latitude>
      <longitude>-3.691163</longitude>
    </element>
  </marcadores>
</root>
```

b) Minifica los datos JSON. ¿Cuántos caracteres ocupan los datos minificados JSON? Minifica los datos XML. ¿Cuántos caracteres ocupan los datos minificados XML?

{"marcadores":[{"latitude":40.416875,"longitude":-3.703308,"city":"Madrid","description":"Puerta delSol"},{"latitude":40.417438,"longitude":-3.693363,"city":"Madrid","description":"Paseo del Prado"},{"latitude":40.407015,"longitude":-3.691163,"city":"Madrid","description":"Estación de Atocha"}]}

JSON: 296 caracteres

<?xml version="1.0" encoding="UTF-8"
?><root><marcadores><latitude>40.416875</latitude><longitude>-3.703308</
longitude><city>Madrid</city><description>Puerta
delSol</description></marcadores><marcadores><latitude>40.417438</latitude><longitude>-
3.693363</longitude><city>Madrid</city><description>Paseo del
Prado</description></marcadores><marcadores><latitude>40.407015</latitude><longitude>-
3.691163</longitude><city>Madrid</city><description>Estación de
Atocha</description></marcadores></root>

XML: 497 caracteres

c) Localiza algún editor y/o parser JSON online

http://json.parser.online.fr/