*Universidad Mariano Gálvez De Guatemala*

*Ing. Wilson Aguín*

*Curso Diseño Y Desarrollo Web*

*Ingeniería En Sistemas De Información*

*Y Ciencias De La Comunicación*

**

*Web Services Soap, Xml, Json, Svn,*

*Git e Integración Continua*

*Oscar David Tizol*

*Carnet 2990-14-1638*

*Semestre 2, 8vo. Ciclo*

*Santa Lucia Cotzumalguapa, Octubre 6 Del 2,018*

***Web Services***

***SOAP*** *(originalmente las siglas de Simple Object Access Protocol) es un* [*protocolo*](https://es.wikipedia.org/wiki/Protocolo_(inform%C3%A1tica))[*estándar*](https://es.wikipedia.org/wiki/Norma_(tecnolog%C3%ADa)) *que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos* [*XML*](https://es.wikipedia.org/wiki/XML)*. Básicamente SOAP es un paradigma de mensajería de una dirección sin estado, que puede ser utilizado para formar protocolos más complejos y completos según las necesidades de las aplicaciones que lo implementan. Es uno de los protocolos utilizados en los* [*servicios Web*](https://es.wikipedia.org/wiki/Servicios_Web)*.*

***Características***

*El protocolo SOAP tiene tres características principales:*

* ***Extensibilidad*** *(seguridad y WS-routing son extensiones aplicadas en el desarrollo).*
* ***Neutralidad*** *(bajo protocolo de transporte* [*TCP*](https://es.wikipedia.org/wiki/Transmission_Control_Protocol) *puede ser utilizado sobre cualquier protocolo de aplicación como* [*HTTP*](https://es.wikipedia.org/wiki/HTTP)*,* [*SMTP*](https://es.wikipedia.org/wiki/SMTP) *o* [*JMS*](https://es.wikipedia.org/wiki/JMS)*).*
* ***Independencia*** *(permite cualquier modelo de programación).*

## ***Estructura del mensaje***

*Un mensaje SOAP es un documento XML ordinario con una estructura definida en la especificación del protocolo. Dicha estructura la conforman las siguientes partes:*

* ***Sobre (Envelope)****: el cual define qué hay en el mensaje y cómo procesarlo, raíz que la estructura*

*Incluye los dos sub-elementos, un Header (Encabezado) y un Body (Cuerpo).*

* ***Header****: esta parte es un mecanismo de extensión ya que permite enviar información relativa a cómo debe ser procesado el mensaje. Es una herramienta para que los mensajes puedan ser enviados de la forma más conveniente para las aplicaciones. El elemento "Header" se compone a su vez de* ***"Header Blocks"*** *que delimitan las unidades de información necesarias para el header.*
* ***Body (obligatoria)****: contiene la información relativa a la llamada y la respuesta*

*Cuando usted envía un mensaje SOAP, lo hace con un motivo en mente. Usted trata de decirle al destinatario que haga algo, o trata de impartir información al servidor. Esta información se llama "carga útil".*

*La carga útil va en el Body del Envolope. También tiene su propio espacio de nombres, en este caso el que corresponde al sistema de gestión de contenidos. En este caso, la elección del espacio de nombres, es completamente arbitraria.*

* ***Fault****: bloque que contiene información relativa a errores que se hayan producido durante el procesado del mensaje y el envío desde el "SOAP Sender" hasta el "Ultimate SOAP Receiver".*
* ***Conjunto de reglas de codificación*** *para expresar instancias de tipos de datos.*
* *Dicho de manera sencilla, predominan dos estilos de programación diferentes para los mensajes SOAP. El primero es el estilo RPC, que se basa en el concepto de usar mensajes SOAP para crear Remote Procedure Calls (Llamadas a procedimientos remotos).*

*En este estilo, la idea es que usted enviará un comando al servidor, como por ejemplo “agregar un artículo”, e incorporará los parámetros para ese comando.*

* ***La Convención*** *para representar llamadas a procedimientos y respuestas.*

***XML*** *es un lenguaje de “marcado ampliable”, lo que significa que proporciona una forma de suministrar información adicional sobre el contenido. Esta información tiene la forma de “tags” (“etiquetas”), las que denotan “elements” (“elementos”).*

*XML Ante todo, es texto. Eso lo hace legible por casi cualquiera, o por casi cualquier cosa. Las etiquetas están indicadas con > y <, con una etiqueta “open” (de apertura) que contiene un nombre, y atributos posibles, como la ID del artículo, y una etiqueta de cierre con una barra (/). Los elementos deben estar auto-contenidos y anidados correctamente.*

*Utilidades*

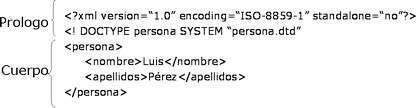
* *Guardar datos*
* *Servicios web*
* *Podcast y suscripción de datos*
* *RDF, micro formatos y tecnologías semánticas*
* *AJAX*
* *Mashups*

*Ventajas*

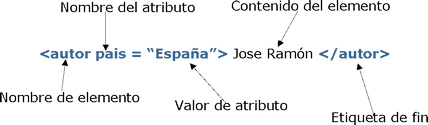
* *Separa datos de presentación*
* *Simplifica compartir e intercambiar datos*
* *La información se almacena en texto plano: software y hardware independiente.*
* *Simplifica el transporte de la información.*
* *Simplifica los cambios de plataforma*
* *Hace los datos disponibles más fácil*
* *Usado en los nuevos lenguajes de internet: xhtml, servicios web, rss, rdf, smil*

## *Estructura de un documento XML*

*Un documento XML está formado por* ***datos de caracteres*** *y* ***marcado****, el marcado lo forman las etiquetas:*

**

## *Estructura*

**

## ***Componentes de un documento XML***

*En un documento XML existen los siguientes componentes:*

* ***Elementos****: Pieza lógica del marcado, se representa con una cadena de texto(dato) encerrada entre etiquetas. Pueden existir elementos   
  vacíos (<br/>). Los elementos pueden contener atributos.*
* ***Instrucciones****: Ordenes especiales para ser utilizadas por la aplicación que procesa <?xml-stylesheet type=“text/css” href=“estilo.css”>*
* ***Las instrucciones XML****. Comienzan por <? Y terminan por ?>.*
* ***Comentarios****: Información que no forma parte del documento. Comienzan por <!-- y terminan por -->.*
* ***Declaraciones de tipo****: Especifican información acerca del documento: <!DOCTYPE persona SYSTEM “persona.dtd”>*
* ***Secciones CDATA****: Se trata de un conjunto de caracteres que no deben ser interpretados por el procesador: <![CDATA[ Aquí se puede meter cualquier carácter, como <, &, >, ... Sin que sean interpretados como marcación]]>*

*JSON**(JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Es un subconjunto de la notación literal de objetos de* [*JavaScript*](https://es.wikipedia.org/wiki/JavaScript) *aunque hoy, debido a su amplia adopción como alternativa a* [*XML*](https://es.wikipedia.org/wiki/XML)*, se considera un formato de lenguaje independiente.*

*Modelos de procesamiento*

*Al ser JSON un formato muy extendido para el intercambio de datos, se han desarrollado API para distintos lenguajes (por ejemplo* [*ActionScript*](https://es.wikipedia.org/wiki/ActionScript)*,* [*C*](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C)*,* [*C++*](https://es.wikipedia.org/wiki/C%2B%2B)*,* [*C#*](https://es.wikipedia.org/wiki/C_Sharp)*,* [*ColdFusion*](https://es.wikipedia.org/wiki/ColdFusion)*,* [*Common Lisp*](https://es.wikipedia.org/wiki/Common_Lisp)*,* [*Delphi*](https://es.wikipedia.org/wiki/Embarcadero_Delphi)*,* [*E*](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_E)*,* [*Eiffel*](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Eiffel)*,* [*Java*](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java)*,* [*JavaScript*](https://es.wikipedia.org/wiki/JavaScript)*,* [*ML*](https://es.wikipedia.org/wiki/ML_(lenguaje_de_programaci%C3%B3n))*,* [*Objective-C*](https://es.wikipedia.org/wiki/Objective-C)*,* [*Objective CAML*](https://es.wikipedia.org/wiki/Ocaml)*,* [*Perl*](https://es.wikipedia.org/wiki/Perl)*,* [*PHP*](https://es.wikipedia.org/wiki/PHP)*,* [*Python*](https://es.wikipedia.org/wiki/Python)*,* [*Rebol*](https://es.wikipedia.org/w/index.php?title=Rebol&action=edit&redlink=1)*,* [*Ruby*](https://es.wikipedia.org/wiki/Ruby)*,* [*Lua*](https://es.wikipedia.org/wiki/Lua) *y* [*Visual FoxPro*](https://es.wikipedia.org/wiki/Visual_FoxPro)*) que permiten analizar sintácticamente, generar, transformar y procesar este tipo de dato.*

*Los modelos de programación más utilizados para tratar con JSON en los distintos lenguajes son:*[*4*](https://es.wikipedia.org/wiki/JSON#cite_note-4)*​*

* *Modelo de objeto.- El JSON completo es almacenado en memoria en un formato de árbol. Este árbol es navegado, analizado y modificado con las API apropiadas. Como lo carga todo en memoria y luego lo procesa este modelo consume muchos recursos. Sin embargo es muy flexible para manipular el contenido. Este modelo es permitido por ejemplo en Java por la JSR 353 y por la biblioteca* [*Jackson*](https://es.wikipedia.org/w/index.php?title=Jackson_(Java)&action=edit&redlink=1)*.*
* *Modelo de flujo: Los datos son leídos o escritos en bloques. Por ejemplo, cada vez que se lee un bloque, el analizador genera eventos apropiados para indicar el tipo de bloque de que se trata. El cliente puede procesar el contenido escuchando los eventos apropiados. Además es el cliente el que decide como se va leyendo el JSON permitiendo parar o saltar contenidos en mitad del proceso. El proceso de escritura tiene propiedades análogas. Por ejemplo este modelo es permitido en java por la JSR 353.*
* *Convirtiendo los objetos JSON en objetos del lenguaje.*

*Sintaxis JSON*

*La sintaxis JSON está basada en los siguientes principios:*

* *los datos consisten en pares de nombre/valor,*
* *los datos están separados por comas,*
* *los objetos están definidos por corchetes {},*
* *los arrays están definidos por corchetes [ ].*

# *Git y SVN*

*Control de versiones fueron creados con el fin de detectar cambios en los documentos o archivos y se encargan de guardar todas las versiones anteriores, incluyendo el registro de fecha y hora, así como el identificador del usuario de un archivo para que los datos puedan ser recuperados y restaurados en cualquier momento. De esta forma, es posible determinar qué usuario ha realizado cambios en un punto determinado. Los objetivos generales de este tipo de sistemas consisten en* ***coordinar el acceso compartido de varios usuarios a los archivos*** *y permitir el desarrollo simultáneo de varias bifurcaciones o branches.*

*El servicio de alojamiento basado en la web para proyectos Git es GitHub, mientras que en  RiouxSVN aloja a Subversion. Proveedores como SourceForge ofrecen alojamiento para ambos sistemas.*

*SVN se basa en un sistema de* ***control de versiones centralizado****. Esto significa que existe un almacén central de datos (el repositorio) accesible a todos los usuarios. Dado que los cambios realizados no pueden ser fusionados entre sí, el sistema evita que dos usuarios puedan editar un mismo archivo al mismo tiempo. El proceso es muy simple, cuando uno de los usuarios accede a un archivo, el sistema lo marca automáticamente como de solo lectura para los demás. Además, Apache Subversion ofrece la posibilidad de descargar y editar directorios individuales sin depender del árbol general de directorios.*

*GIT: es un* ***sistema de control de versiones distribuido****, lo que significa que, aunque existe un repositorio central en el cual se incorporan los cambios, todos los usuarios pueden descargar su propia copia de trabajo. De esta forma, todos tienen acceso al repositorio completo, incluyendo el historial local, sin depender de ningún tipo de conexión de red. Todos los cambios se transfieren rápidamente al repositorio central. Como consecuencia, Git no ofrece ningún sistema de bloqueo, sino que cada usuario genera sus propios directorios o branches dentro del árbol para ser cargados posteriormente al repositorio central. Por defecto, cada usuario tiene permisos de lectura y escritura para los diferentes directorios (en caso de que se quiera asignar permisos especiales, será necesario crear otros directorios raíz).*

# ***INTEGRACIÓN CONTINUA***

*La integración continua (continuous integration en* [*inglés*](https://es.wikipedia.org/wiki/Idioma_ingl%C3%A9s)*) es un modelo* [*informático*](https://es.wikipedia.org/wiki/Inform%C3%A1tico) *que consiste en hacer integraciones automáticas de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes. Entendemos por integración la* [*compilación*](https://es.wikipedia.org/wiki/Compilaci%C3%B3n) *y* [*ejecución de pruebas*](https://es.wikipedia.org/wiki/Testeo_de_software) *de todo un proyecto.*

*El proceso suele ser: cada cierto tiempo (horas), descargarse las fuentes desde el* [*control de versiones*](https://es.wikipedia.org/wiki/Control_de_versiones) *(por ejemplo* [*CVS*](https://es.wikipedia.org/wiki/CVS)*,* [*Git*](https://es.wikipedia.org/wiki/Git)*,* [*Subversion*](https://es.wikipedia.org/wiki/Subversion)*,* [*Mercurial*](https://es.wikipedia.org/wiki/Mercurial) *o* [*Microsoft Visual SourceSafe*](https://es.wikipedia.org/wiki/Microsoft_Visual_SourceSafe)*) compilarlo, ejecutar pruebas y generar informes.*

*A menudo la integración continua está asociada con las metodologías de* [*programación extrema*](https://es.wikipedia.org/wiki/Programaci%C3%B3n_extrema) *y* [*desarrollo ágil*](https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software)*.*

*Ventajas*

* *Los desarrolladores pueden detectar y solucionar problemas de integración de forma continua, evitando el caos de última hora cuando se acercan las fechas de entrega.*
* *Disponibilidad constante de una versión para pruebas, demos o lanzamientos anticipados.*
* *Ejecución inmediata de las pruebas unitarias.*
* *Monitorización continúa de las métricas de calidad del proyecto.*