DIPLOMADO

# Desarrollo de sistemas con tecnología Java

## Módulo 10

API RESTful con Spring Boot

Material didáctico: Web Service, API REST Y SOAP

*M. En C. Jesús Hernández Cabrera*

# Índice.

## Objetivos.

En este documento se estudian y discuten algunos de los paradigmas actuales para la integración de aplicaciones distribuidas, con un enfoque en los *web services* SOAP y las API REST. El objetivo es proporcionar al alumno un contexto sólido para mejorar su comprensión y aprovechamiento en la programación de sistemas API RESTful utilizando el Framework Spring Boot.

## Fundamentos de servicios Web.

Los *Web services* son una arquitectura de software que habilita la intercomunicación de sistemas por medio de una interface general utilizando los protocolos de internet HTTP(Hyper Text Transfer Protocol), los web services emplean de forma predilecta del formato XML (Extensible Markup Languague), sin embargo hay una alternativa mas simple empleando formato JSON(JavaScript Object Notation) y simplifican en gran medida el desarrollo de aplicaciones distribuidas.

Los servicios web apoyan a la interoperabilidad de los sistemas dentro de las mismas organizaciones y sobre todo la interoperabilidad con organizaciones externas, debido a lo antes mencionado, es importante comentar que el protocolo de comunicación estandarizado (HTTP) y el formato de archivos de intercambio (XML) (Coulouris, Dollimore, Kindberg, & Blair, 2012) son partes esenciales de este estilo arquitectónico.

Los web services regresan al modelo cliente-servidor, pero es esencial distinguir entre un servidor web y un web service. Mientras que un servidor web procesa y responde solicitudes HTTP, su propósito principal es servir contenido estático o dinámico, como páginas web o documentos generados dinámicamente mediante tecnologías como Spring o PHP. En contraste, un web service no solo se limita a entregar contenido, sino que ofrece un conjunto de funcionalidades específicas a través de su interfaz definida.

Esta interfaz está diseñada para facilitar la interacción entre diferentes aplicaciones, permitiendo el intercambio de datos y la ejecución de operaciones más complejas. Los web services permiten realizar transacciones, procesar datos o ejecutar funciones de negocio **entre sistemas**, y su diseño se adapta para cumplir con las **necesidades específicas** del cliente. De este modo, van más allá de la simple entrega de contenido, habilitando la comunicación y colaboración entre plataformas distintas mediante estándares abiertos.

Existen varios protocolos que se pueden utilizar para implementar web services, cada uno con sus propias características y ventajas. Los más comunes son SOAP (Simple Object Access Protocol) y REST (Representational State Transfer). SOAP es un protocolo más estructurado y formal que utiliza XML para el intercambio de mensajes y es ideal para escenarios donde se requieren niveles elevados de seguridad, transacciones complejas o compatibilidad con servicios heredados. Por otro lado, REST es un enfoque más ligero y flexible que aprovecha completamente los métodos HTTP y es capaz de trabajar con diferentes formatos de datos, como JSON o XML. A continuación, se explicarán en detalle ambos protocolos, sus diferencias y cómo se aplican en distintos entornos.

## El formato JSON

El formato JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos, fácil de leer y escribir para los humanos, y sencillo de interpretar y generar para las máquinas (JSON.org). Está basado en la sintaxis de objetos de JavaScript, pero es independiente del lenguaje, lo que lo convierte en una opción popular para la transferencia de datos entre sistemas. En JSON, los datos se estructuran en pares clave-valor, lo que facilita su organización y manipulación. Es ampliamente utilizado en APIs y web services debido a su simplicidad y eficiencia. Si eres nuevo en JSON te recomiendo iniciar directamente en la página de JSON.ORG(https://www.json.org/json-es.html)

```
{
  "nombre": "Juan",
  "edad": 30,
  "ciudad": "Ciudad de México"
}
```

Código 1.- Ejemplo de un Objeto JSON.

## El formato XML

El formato XML (eXtensible Markup Language) es un estándar de marcado diseñado para almacenar y transportar datos de manera estructurada y legible tanto para humanos como para máquinas. Utiliza una estructura de etiquetas anidadas que permite organizar los datos de manera jerárquica. Es altamente extensible, lo que significa que los usuarios pueden definir sus propias etiquetas según sea necesario. XML ha sido ampliamente utilizado en web services, documentos de configuración y diversas aplicaciones donde se necesita intercambiar o almacenar datos de manera flexible y compatible con múltiples sistemas como a continuación se explica el protocolo SOAP.

```
<persona>
  <nombre>Juan</nombre>
  <edad>30</edad>
  <ciudad>Ciudad de México</ciudad>
</persona>
```

Código 2.- Ejemplo de un Objeto XML.

## Estructura de un Web service SOAP.

El formato XML se utiliza comúnmente para el intercambio de información en web services. La información se encapsula y se gestiona utilizando protocolos de transmisión como HTTP y SOAP. El protocolo SOAP (Simple Object Access Protocol) define cómo los mensajes deben ser estructurados y enviados, asegurando que las aplicaciones puedan comunicarse correctamente, incluso si están escritas en diferentes lenguajes o corren en distintas plataformas. En la siguiente imagen se muestra toda la infraestructura de un web service, incluyendo la interacción entre cliente, servidor y los protocolos involucrados.

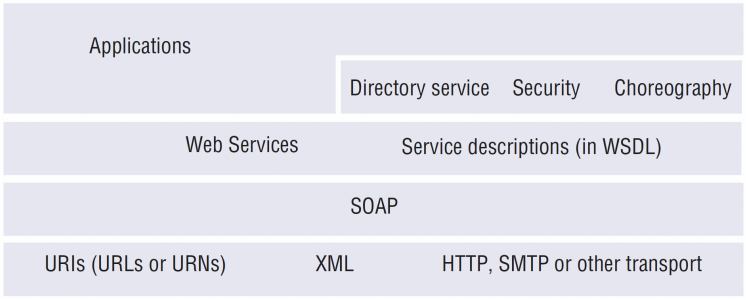


Ilustración 1.- Estructura de los web services. (Coulouris, Dollimore, Kindberg, & Blair, 2012)

Cómo se puede observar en la parte superior se encuentran implementadas las aplicaciones y los servicios web, junto con componentes de software de soporte adicionales. En la siguiente capa hacia abajo se encuentran los componentes que describen las interfaces del servicio, en especial el Web Service Description Language (WSDL), adicionalmente describe dos características:

- La URI del servicio.
- Métodos de comunicación.

El WSDL es una especificación disponible en (W3C, The World Wide Web Consortium (W3C), 2007), el cuál define un lenguaje en formato XML con elementos para describir servicios web y sus partes.

En La sección 2 de está especificación, de nombre “Componentes del modelo” (Components model ) incluye los siguientes apartados para la descripción de los servicios: Interface, Interface Fault, Interface Operation, Interface Meessage Reference, Interface Fault Tolerance, Binding, Binding Fault, Binding Message Reference, Service y Endpoint. Esta descripción establece las bases o criterios para el entendimiento entre clientes y servicios, dichas bases están relacionadas a que el servicio debe definir que expone como servicio al cliente y este debe recibir la información necesaria al cliente para que este lo descubra y consuma de forma adecuada. En la ilustración 2 se puede observar la estructura principal de los elementos para la descripción de esta información.

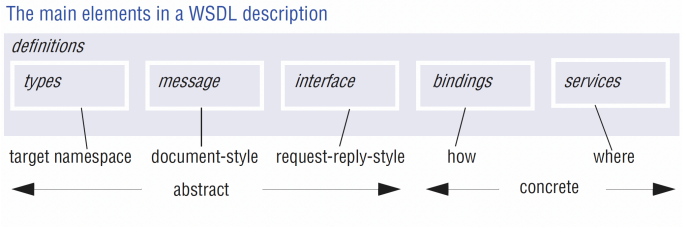


Ilustración 2.- Estructura principal del lenguaje WSDL

En donde se destacan los componentes destino de cada parte WSDL, por ejemplo, para indicar cual es el mecanismo de cómo se recibirán las peticiones y como estas serán contestadas. La sección *interface* debe proporcionar esa información a través de un documento XML como el siguiente (W3C, The World Wide Web Consortium (W3C), 2007):

```
<description>
  <interface
        name="xs:NCName"
        extends="list of xs:QName"?
        styleDefault="list of xs:anyURI"? >
    <documentation />*
    [ <fault /> | <operation /> ]*
  </interface>
</description>
```

Ilustración 3.- Descripción de la interface de un Web Service.

El mismo documento describe la colección de operaciones que el servicio proveerá, estas operaciones indican el patrón de intercambio de los mensajes entre las partes (Servidor y cliente), estos patrones están resumidos en la tabla de la siguiente imagen.

| | | | | |
|---|---|---|---|---|
| In-Out | *Request* | *Reply* | | May replace *Reply* |
| In-Only | *Request* | | | No fault message |
| Robust In-Only | *Request* | | Guaranteed | May be sent |
| Out-In | *Reply* | *Request* | | May replace *Reply* |
| Out-Only | | *Request* | | No fault message |
| Robust Out-Only | | *Request* | Guaranteed | May send fault |

Ilustración 4.- Tipos de mensajes y su descripción dentro de SWDL.

Un componente clave en la estructura del WSDL es la sección de configuración de la vinculación de componentes (Binding). Según la especificación oficial (W3C, 2007), esta se describe como: 'Un componente de vinculación que especifica un formato de mensaje concreto y un protocolo de transmisión que se puede utilizar para definir un punto final'. Esta vinculación es crucial para establecer cómo se deben estructurar los mensajes y a través de qué protocolo se transmitirán entre el cliente y el servicio web.

Describir servicios web mediante WSDL puede ser una tarea tediosa debido a su sintaxis basada en XML, como se mencionó anteriormente. Por esta razón, existen herramientas que automatizan gran parte de este proceso. Entre las más utilizadas están WSDL4 y SOAP UI, las cuales simplifican considerablemente la creación y configuración de estos servicios, permitiendo a los desarrolladores enfocarse en la funcionalidad sin preocuparse por los detalles de bajo nivel.

# SOAP y REST.

## *Qué es SOAP.*

SOAP es el acrónimo de Simple Object Access Protocol, en este caso es una especificación para la definición de servicios web y específicamente para la parte de especificación de como la información se intercambia. Su documentación oficial se puede consultar en (W3C, W3C, 2007) , en el cual se explica que se trata de un framework ligero de mensajería extensible con el objetivo de funcionar bajo diferentes protocolos de comunicación en un entorno distribuido y descentralizado.

Dentro de las principales características destacan que emplea la sintaxis XML para la definición del framework, lo que abona en gran medida a la característica de independencia de los mensajes entre lenguajes de programación o semánticas específicas.

SOAP especifica dentro de (W3C, W3C, 2007) cómo usar el método POST del protocolo HTTP, con el objetivo de coordinar o admitir la comunicación entre el cliente y el servidor en combinación del formato XML de los mensajes. Estos mensajes son definidos por un documento XML, un ejemplo se observa en la siguiente imagen:

Example 1: SOAP message containing a SOAP header block and a SOAP body

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
   <n:priority>1</n:priority>
   <n:expires>2001-06-22T14:00:00-05:00</n:expires>
  </n:alertcontrol>
 </env:Header>
 <env:Body>
  <m:alert xmlns:m="http://example.org/alert">
   <m:msg>Pick up Mary at school at 2pm</m:msg>
  </m:alert>
 </env:Body>
</env:Envelope>
```

Ilustración 5.- Ejemplo de un mensaje SOAP. (W3C, W3C, 2007)

El principal objetivo de SOAP es facilitar el acceso a datos utilizando un protocolo conocido y de fácil uso, como los protocolos de Internet, en particular HTTP y sus métodos, como el método POST. SOAP es un estándar desarrollado en colaboración con Microsoft, cuyo propósito es proporcionar un marco para el intercambio de información entre aplicaciones en entornos empresariales.

SOAP debe emplearse cuando el entorno del problema requiere cumplir con especificaciones formales para la lógica de negocio entre clientes y servidores. Además, es recomendable optar por SOAP en casos donde los requisitos de seguridad sean estrictos, ya que este protocolo permite definir restricciones de autenticación y autorización a través de WSDL. Finalmente, SOAP es útil para establecer un mecanismo estandarizado que permita el acceso a objetos en servidores de aplicaciones, garantizando una interacción fiable y segura.

Por otra parte se debe mencionar que SOAP tiene sus desventajas, una de ellas es la complejidad de su definición, al estar basada en XML, la descripción de sus características puede ser complicada y los archivos descriptores de mansaje pueden ser difíciles de leer para los programadores. Además de lo anterior, SOAP consume mucho ancho de banda por el tamaño de los mensajes y no es un mecanismo sencillo de emplear o consumir, para que los desarrolladores puedan usarlo deben tener conocimientos al menos básicos de su implementación.

## *Qué es REST.*

Es el acrónimo de Representacional State Transfer, es una especificación desarrollada como trabajo de disertación de doctorado de Roy Fielding en la Universidad de California (Fielding, 2000), en este documento se describe a esta arquitectura como un estilo arquitectónico para sistemas distribuidos de hiper media, al inicio del capítulo 6 se puede leer lo siguiente*: “Este trabajo se realizó como parte de los esfuerzos de Internet Engineering Taskforce (IETF) y World*

*Wide Web Consortium (W3C) para definir los estándares arquitectónicos para la Web: HTTP, URI y HTML."* Grupo en donde Fielding era integrante y quien consideraba que no se le estaba sacando provecho a estos protocolos, específicamente a los HTTP Methods diferentes al GET y POST: PUT, PATCH, DELETE, entre otros.

Una de sus primeras declaraciones es el establecimiento y separación de las responsabilidades del cliente y el servidor, con el objetivo de mejorar la portabilidad de la interfaz de usuario sin importar la plataforma y además de permitir mejorar la escalabilidad de los sistemas.

En su sección 5.1.3 (Fielding, 2000) describe la naturaleza de del concepto *"sin estado"*(stateless) de esta especificación, lo que significa que el estado debe ser almacenado por completo del lado del cliente y se le debe de dispensar la responsabilidad al servidor. Estas características tienen el objetivo de mejorar la visibilidad, confiabilidad y la escalabilidad de la siguiente manera:

***Visibilidad***.- El servidor solo requiere saber la información propia de las peticiones(request) durante los intercambios de datos, eso permite que la información este separada por contextos de solicitud, sin embargo, hay que asegurar que los datos son suficientes para realizar su tarea. La visibilidad también permite que cada transacción sea independiente, esto es particularmente útil para tareas o actividades de depuración.

***Confiabilidad***.- Esta arquitectura permite la recuperación de fallas parciales durante el intercambio de información. Y a que no depende del estado de una sesión previa o almacenamiento de estado. Esto significa que, si ocurre una falla en la conexión, se puede recuperar fácilmente sin la necesidad de retener información del estado previo, ya que cada solicitud puede comenzar de nuevo de manera independiente.

***Escalabilidad***.- Dado que el servidor no debe almacenar el estado de las conversaciones con el cliente, este puede liberar recursos rápidamente, específicamente esto pasa cuando una petición ha sido respondida por completo. Al liberar recursos inmediatamente después de completar una solicitud, los servidores pueden atender más solicitudes de manera eficiente y sin la sobrecarga de gestionar estados de múltiples clientes simultáneamente. Esto es una de las principales razones por las que REST se utiliza ampliamente en aplicaciones a gran escala.

Otra de las características importantes de este estilo arquitectónico, es el concepto de cache del lado del cliente con el objetivo de mejorar la eficiencia de transferencia vía internet.

En su sección 5.1.5 describe la característica más importante desde mi punto de vista, y esta es la característica de interfaz uniforme, dicha interfaz está diseñada para soportar múltiples formatos de internet, para lo cual la misma debe definirse siguiendo 4 restricciones: identificación de recursos; manipulación de recursos a través de representaciones; mensajes autodescriptivos; e hipermedia como motor de estado de aplicación.

En cuanto al soporte http, el REST establece que se deben realizar las 3 siguientes adecuaciones para desarrollar un sistema de este tipo:

1. La correcta implementación de los métodos del protocolo HTTP : OPTIONS, HEAD, GET, POST, PUT, DELETE. TRACE y CONNECT.

2. Adecuada construcción de las URI (Universal Resource Identifier) con la estructura definida en la especificación estándar que consta de:

   protocolo://host[:puerto]/recurso?parametro=valor&…

3. La implementación de manejo de información de intercambio combinada con URI´s de referencia. Por ejemplo:

   http://www.elgainternet.com/api/lectura/:id

La siguiente tabla establece un resumen de cómo deben responder los diferentes verbos HTTP en un API REST.

| **Método HTTP** | **Operación CRUD** | **Respuestas HTTP Status, Elemento específico: (ejemplo: /users/{id})** | **Respuestas HTTP Status a la colección entera. (ejemplo: /users/ )** |
|---|---|---|---|
| **POST** | Create / insert | 404 | 201<br>Created<br>Res.: el documento con nuevo _id |
| **GET** | Read /Select | 200 OK Resp.: Un sólo documento | /users/: 200 OK, Resp.: Todos los documentos |
| **PUT** | Replace / Update | 200 OK ó 204 (Body sin contenido). Ó 404 (id no encontrado) | 405 acción no permitida. |
| **PATCH** | Update Only | 200 (OK) or 204 (Body sin contenido). 404 (No encontrado) | 405 acción no permitida. |
| **DELETE** | Delete | 200 (OK) ó 404 (No encontrado) | 405 acción no permitida. |

## Comparativa REST vs SOAP

| Característica | SOAP | REST |
|---|---|---|
| **Formatos de intercambio** | Solo XML | XML y JSON |
| **Uso transaccional** | Muy bueno | No recomendable. |
| **Filosofía** | Open Web | Empresarial y formal |
| **Complejidad de implementación** | difícil, requiere de una curva de aprendizaje. | Relativamente fácil de implementar. |
| **Mantenimiento** | Requiere de esfuerzo. | Fácil de mantener. |
| **Seguridad** | Es parte del protocolo. Es buena. | No forma parte del protocolo, se implementa empleando tecnologías adicionales como por ejemplo los Web Tokens. |
| **Eficiencia** | Sobrecargado | Eficiente |
| **Estandarizado** | Si WS-Standards | No |

## Conclusiones

En el contexto de las tecnologías de sistemas distribuidos, tanto la especificación SOAP como el estilo arquitectónico REST API juegan un papel crucial en el diseño de sistemas modernos. SOAP, fue una de las primeras implementaciones que permitió la evolución de los sistemas empresariales, aunque su implementación puede ser compleja.

En la actualidad, REST ha ganado popularidad por simplificar considerablemente el desarrollo de sistemas, ofreciendo una alternativa más ligera y flexible. Sin embargo, esta simplicidad conlleva la pérdida de algunas características importantes que SOAP si proporciona, como sus capacidades avanzadas de seguridad y formalismo en la estructura de mensajes.

Por lo tanto, no se puede afirmar que REST sea la solución adecuada para todos los casos. La elección entre SOAP y REST debe basarse en los requisitos específicos de cada proyecto. Ambos enfoques tienen fortalezas y debilidades, por lo que es fundamental analizar el contexto y las necesidades de la solución para determinar cuál es la opción más adecuada.

## Bibliografía

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). *DISTRIBUTED SYSTEMS Concepts and Design.* USA: Pearson.

Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* California: UNIVERSITY OF CALIFORNIA, Irvine.

JSON.org. (n.d.). *Introducing JSON.* Retrieved from https://www.json.org/json-en.html

RedHat. (n.d.). *Red Hat*. Retrieved from Cloud Computing: https://www.redhat.com/es/topics/cloud-computing/what-is-iaas

W3C. (2007, june 26). *The World Wide Web Consortium (W3C)*. Retrieved from Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language: https://www.w3.org/TR/wsdl/

W3C. (2007, Abril 27). *W3C*. Retrieved from SOAP Version 1.2 Part 1: Messaging Framework (Second Edition): https://www.w3.org/TR/soap12/