# Web Services

**Profesor:**
*Máster Carlos Carranza Blanco*
**Ingeniería del Software**

11/8/2022

# Web Services - Definición

- *Un servicio web (en inglés, Web Service o Web services) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.*

# Web Services - Definición

- *La magia de los Web Services está en que el programador de X puede crear un Web Service para transferir datos sin necesidad de conocer al programador Y, ni a los programas que éste tiene a cargo. De modo que quien quiera recibir los datos solo necesita usar el Web Service y punto.*
- *Esto significa que pueden existir transferencias de datos entre distintas aplicaciones –programas- que funcionan en varios computadores, con distintos sistemas operativos, y que pertenezcan a diferentes empresas o instituciones.*

# Web Services SOAP…

- *El término Web Services describe una forma estandarizada de integrar aplicaciones WEB mediante el uso de XML, SOAP, WSDL y UDDI sobre los protocolos de la Internet. XML es usado para describir los datos, SOAP se ocupa para la transferencia de los datos, WSDL se emplea para describir los servicios disponibles y UDDI se ocupa para conocer cuales son los servicios disponibles.*
- *Uno de los usos principales es permitir la comunicación entre las empresas y entre las empresas y sus clientes.*

# Web Services SOAP…

- *Un servicio Web XML puede ser utilizado internamente por una aplicación o bien ser expuesto de forma externa en Internet por varias aplicaciones.*
- *Podemos definir un servicio Web XML como una clase a la que podemos acceder utilizando estándares de Internet.*
- *¿por qué no podríamos invocar métodos de componentes a través de la red independientemente de dónde se encuentren, del lenguaje en el que estén escritos y de la plataforma de computación en la que se ejecuten?. Esto es precisamente lo que ofrecen los Servicios Web.*

# SOAP

- *La tecnología que está detrás de todo ello se llama SOAP. Este acrónimo (Simple Object Access Protocol) describe un concepto tecnológico basado en la sencillez y la flexibilidad que hace uso de tecnologías y estándares comunes para conseguir las promesas de la ubicuidad de los servicios, la transparencia de los datos y la independencia de la plataforma que según hemos visto, se hacen necesarios en las aplicaciones actuales.*

# Base tecnológica de SOAP

*SOAP utiliza para su implementación tecnologías y estándares muy conocidos y accesibles como son XML o el protocolo HTTP.*

- *Dado que los mensajes entre componentes y los datos de los parámetros para llamadas a métodos remotos se envían en formato XML basado en texto plano, SOAP se puede utilizar para comunicarse con cualquier plataforma de computación.*

- *El uso de HTTP como protocolo principal de comunicación hace que cualquier servidor web del mercado pueda actuar como servidor SOAP, reduciendo la cantidad de software a desarrollar y haciendo la tecnología disponible inmediatamente.*

# Descubrimiento de servicios: WSDL

- *Otro de los estándares que se definen en SOAP es WSDL (Web Service Definition Language). Se trata de un formato estándar para describir las interfaces de los servicios web. WSDL describe qué métodos están disponibles a través de un servicio Web y cuáles son los parámetros y valores devueltos por éstos. Antes de usar un componente que actúa como servicio web se debe leer su archivo WSDL para averiguar cómo utilizarlo.*

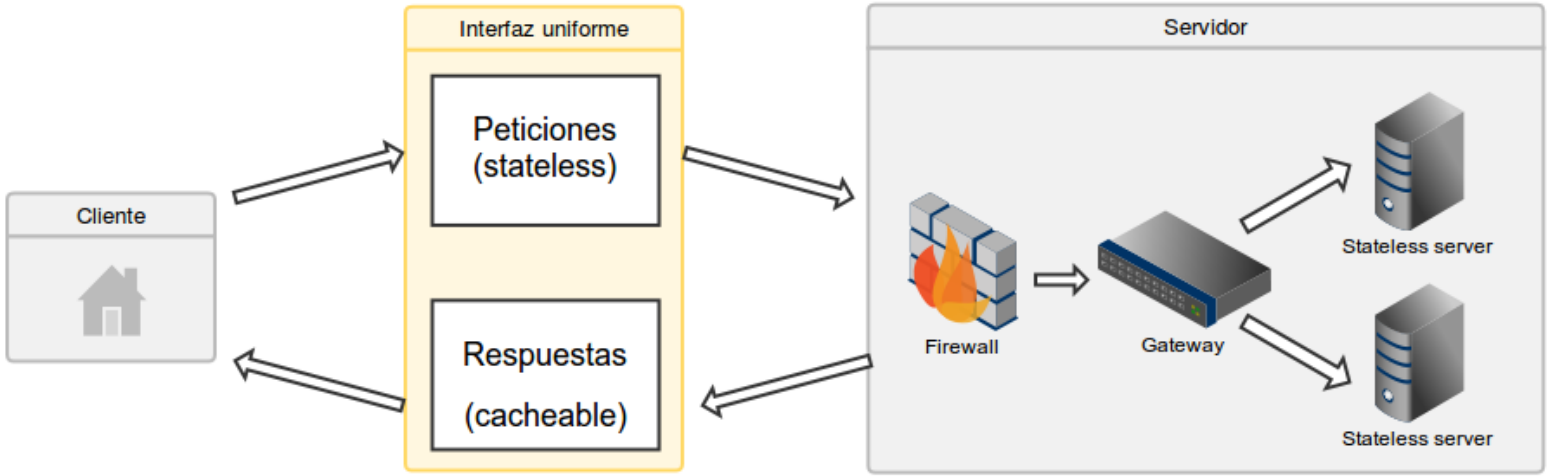
# Web Services RESTful…

- *REST es un estilo de arquitectura que abstrae los elementos de dicha arquitectura dentro de un sistema hypermedia distribuido. Utilizando palabras llanas, podríamos decir que REST es un conjunto de principios, o maneras de hacer las cosas, que define la interacción entre distintos componentes, es decir, las reglas que dichos componentes tienen que seguir. El protocolo más usado que cumple esta definición, es el protocolo HTTP.*

# Reglas de arquitectura REST

- ***Arquitectura cliente-servidor***.
- ***Stateless***: *esto significa que nuestro servidor no tiene porqué almacenar datos del cliente para mantener un estado del mismo.*
- ***Cacheable***: *esta norma implica que el servidor que sirve las peticiones del cliente debe definir algún modo de cachear dichas peticiones, para aumentar el rendimiento, escalabilidad, etc.*
- ***Sistema por capas***.
- ***Interfaz uniforme***: *esta regla simplifica el protocolo y aumenta la escabilidad y rendimiento del sistema. No queremos que la interfaz de comunicación entre un cliente y el servidor dependa del servidor al que estamos haciendo las peticiones, ni mucho menos del cliente.*

# Reglas de arquitectura REST



# Web Services RESTful…

- *Toda aplicación REST debe poder identificar sus recursos de manera uniforme. HTTP implementa esto usando las llamadas URIs (Uniform resource identifier).*
- *Un servicio web RESTful hace referencia a un servicio web que implementa la arquitectura REST.*
- *Un servicio web RESTful contiene lo siguiente:*
  - ***URI del recurso**. Por ejemplo:*

    *http://api.servicio.com/recursos/casas/1 (esto nos daría acceso al recurso "Casa" con el ID "1")*

# Web Services RESTful…

- ***El tipo de la representación de dicho recurso***. *Por ejemplo, podemos devolver en nuestra cabecera "Content-type: application/json", por lo que el cliente sabrá que el contenido de la respuesta es una cadena en formato JSON, y podrá procesarla como prefiera. El tipo es arbitrario, siendo los más comunes JSON, XML y TXT.*

- ***Operaciones soportadas***: *HTTP define varios tipos de operaciones (verbos), que pueden ser GET, PUT, POST, DELETE, PURGE, entre otros. Es importante saber para que están pensados cada verbo, de modo que sean utilizados correctamente por los clientes.*

# Web Services RESTful…

- ***Hipervínculos***: *por último, nuestra respuesta puede incluir hipervínculos hacia otras acciones que podamos realizar sobre los recursos. Normalmente se incluyen en el mismo contenido de la respuesta, así si por ejemplo, nuestra respuesta es un objeto en JSON, podemos añadir una propiedad más con los hipervínculos a las acciones que admite el objeto.*

# REST VS. SOAP

REST (Representational State Transfer) y SOAP (Simple Object Access Protocol) representan dos estilos bastante diferentes para implementar una Arquitectura de Servicios Distribuidos.

- **REST** es un patrón de arquitectura construido con verbos simples que encajan perfectamente con HTTP. Si bien, aunque los principios de Arquitectura de REST podrían aplicarse a otros protocolos adicionales a HTTP, en la práctica, las implementaciones de REST se basan completamente en HTTP.
- **SOAP** es un protocolo de mensajería basado en XML (mensajes SOAP con un formato XML concreto) que puede utilizarse con cualquier protocolo de comunicaciones (Transporte), incluido el propio HTTP.

# REST VS. SOAP

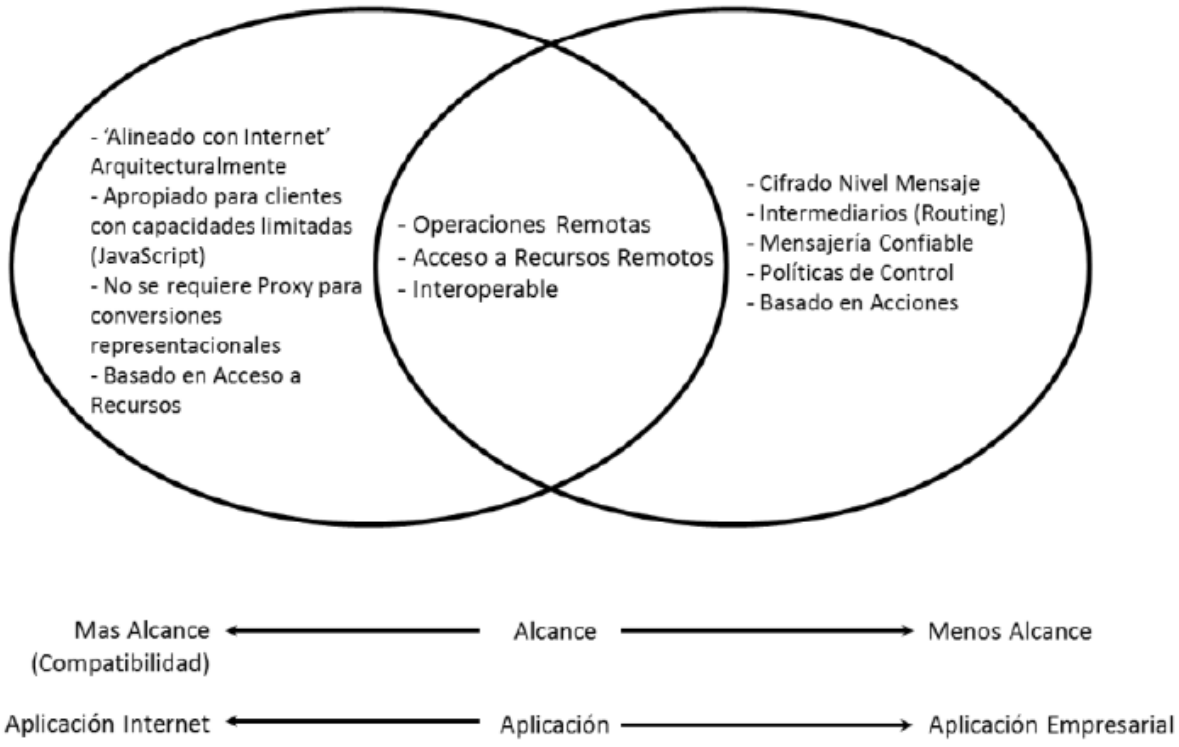



**Figura 66.- Escenarios REST o SOAP**

# Ventajas SOAP

- Bueno para datos (Las comunicaciones son estrictas y estructuradas).
- Dispone de proxies fuertemente tipados gracias a WSDL.
- Funciona sobre diferentes protocolos de comunicaciones. El uso de otros protocolos no HTTP (como TCP, NamedPipes, MSMQ, etc.), puede mejorar rendimiento en ciertos escenarios.

# Desventajas SOAP

- Los mensajes de SOAP no son “cacheables”.
- No se puede hacer uso de mensajes SOAP en JavaScript (Para AJAX debe utilizarse REST).

# Ventajas REST

- Gobernado por las especificaciones HTTP por lo que los servicios actúan como Recursos, igual que Imágenes o documentos HTML.
- Los Datos pueden bien mantenerse de forma estricta o de forma desacoplada (no tan estricto como SOAP).
- Los Recursos REST pueden consumirse fácilmente desde código JavaScript (AJAX, etc.)
- Los mensajes son ligeros, por lo que el rendimiento y la escalabilidad que ofrece es muy alta. Importante para Internet.
- REST puede utilizar bien XML o JSON como formato de los datos.

# Desventajas REST

- Es difícil trabajar con objetos fuertemente tipados en el código del servidor, aunque esto depende de las implementaciones tecnológicas y está mejorando en las últimas versiones.
- Solo funciona normalmente sobre HTTP (No es bueno para aplicaciones de alto rendimiento y tiempo real, como aplicaciones de bolsa, etc.)
- Las llamadas a REST están restringidas a Verbos HTTP (GET, POST, PUT, DELETE, etc.)

# Conclusión SOAP vs REST

- REST es normalmente más adecuada para Servicios Distribuidos accesibles públicamente (Internet) o en casos en los que un Servicio puede ser accedido por consumidores desconocidos.

- SOAP se ajusta, por el contrario, mucho mejor a implementar rangos de implementaciones procedurales, como un interfaz entre las diferentes Capas de una Arquitectura de aplicación o en definitiva, Aplicaciones Empresariales privadas.

  Con SOAP no estamos restringidos a HTTP. Las especificaciones estándar WS-*, que pueden utilizarse sobre SOAP, proporcionan un estándar y por lo tanto un camino interoperable para trabajar con aspectos avanzados como SEGURIDAD, TRANSACCIONES, DIRECCIONAMIENTO y FIABILIDAD.

https://www.nginx.com/blog/time-to-move-to-a-four-tier-application-architecture/