2018

**Optativo 1 y 2**

SISTEMAS DISTRIBUIDOS

PEDRO GIMÉNEZ ALDEGUER 15419933C

Contenido:

[**1.** **Websockets:** 2](#_Toc529824443)

[ **¿Qué son?:** 2](#_Toc529824444)

[ **¿Para qué sirven?:** 2](#_Toc529824445)

[ **Relación con las tecnologías vistas anteriormente:** 2](#_Toc529824446)

[ **Ejemplos de frameworks y librerias:** 3](#_Toc529824447)

[ **Compatibilidades:** 4](#_Toc529824448)

[ **Bibliografía:** 5](#_Toc529824449)

[**2.** **WSDL:** 5](#_Toc529824450)

[ **Definición:** 5](#_Toc529824451)

[ **Analizar los modelos WSDL:** 5](#_Toc529824452)

[ **WS-I:** 6](#_Toc529824453)

[ **¿Cómo afecta la generación del stub que enviará los mensajes SOAP?:** 6](#_Toc529824454)

[ **Bibliografía:** 6](#_Toc529824455)

# **Websockets:**

## **¿Qué son?:**

**WebSockets** es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP. Hace posible abrir una sesión de comunicación interactiva entre el navegador del usuario y un servidor. Con esta API, puede enviar mensajes a un servidor y recibir respuestas controladas por eventos sin tener que consultar al servidor para una respuesta.

Debido a que las conexiones TCP comunes sobre puertos diferentes al 80 son habitualmente bloqueadas por los administradores de redes, el uso de esta tecnología proporcionaría una solución a este tipo de limitaciones proveyendo una funcionalidad similar a la apertura de varias conexiones en distintos puertos, pero multiplexando diferentes servicios WebSocket sobre un único puerto TCP (a costa de una pequeña sobrecarga del protocolo).

## **¿Para qué sirven?:**

WebSocket es una tecnología que nos permite crear sistemas bidireccionales de comunicación instantánea y simultánea entre clientes y servidores. WebSocket resuelve el problema permitiendo que la información se transmita a través de mensajes en los dos sentidos en una misma conexión.

Esto nos trae muchas ventajas, como la reducción de carga a través de la red y principalmente una arquitectura que nos permite diseñar nuestros proyectos de una manera más sencilla, minimizando los problemas que nos podemos encontrar. Ahora bien, suele ser muy conveniente utilizar abstracciones que trabajan encima de WebSocket al desarrollar nuestras aplicaciones, ya que ellas nos proveen de utilidades ya pensadas para resolver los problemas más comunes a ocurrir.

## **Relación con las tecnologías vistas anteriormente:**

¿Complementaria? ¿Alternativa?

**Modelo Cliente/Servidor:** un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta.

**Modelo CGI:** es una importante tecnología de la World Wide Web que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa cuyo resultado final de la ejecución son objetos MIME. Las aplicaciones que se ejecutan en el servidor reciben el nombre de **CGIs**.

**Aplicación XHTML:** es básicamente HTML expresado como XML válido. Es más estricto a nivel técnico, pero esto permite que posteriormente sea más fácil al hacer cambios o buscar errores entre otros.

**Applets de Java:** es un applet escrito en el lenguaje de programación Java. Los applets de Java pueden ejecutarse en un navegador web utilizando la Java Virtual Machine (JVM), o en el AppletViewer de Sun.

**Componentes ActiveX:** es un entorno para definir componentes de software reusables de forma independiente del lenguaje de programación. Las aplicaciones de software pueden ser diseñadas por uno o más de esos componentes para así proveer su correspondiente funcionalidad.

**Modelo Servlets:** es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor. Aunque los servlets pueden responder a cualquier tipo de solicitudes, estos son utilizados comúnmente para extender las aplicaciones alojadas por servidores web, de tal manera que pueden ser vistos como applets de Java que se ejecutan en servidores en vez de navegadores web.

**Modelo ISAPI, NSAPI:**

**Modelo de Páginas Activas:** Se basa en fragmentos de código insertados en páginas html que el servidor se encarga de interpretar antes de servir dichas páginas. Se realiza con el fin de dinamizar las páginas. Esta es la forma de desarrollar más común actualmente, en parte por la amplia alternativa de lenguajes dependientes para utilizar con la plataforma.

**Middleware:** es un *software* que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, o paquetes de programas, redes, *hardware* y/o sistemas operativos. Éste simplifica el trabajo de los programadores en la compleja tarea de generar las conexiones y sincronizaciones que son necesarias en los sistemas distribuidos. De esta forma, se provee una solución que mejora la calidad de servicio, así como la seguridad, el envío de mensajes, la actualización del directorio de servicio, etc.

## **Ejemplos de frameworks y librerias:**

A continuación, voy a explicar algunos ejemplos de frameworks y librerias para WebSockets:

**1. Socket.io:**

WebSocket contenedor / API ampliamente popular y usado de 2018. Consistente en un servidor Node.js y una biblioteca de cliente Javascript, socket.io proporciona confiabilidad para el manejo de proxies y balanceadores de carga, así como firewall personal y software antivirus, e incluso admite transmisión binaria.

**2. WS:**

WS es un popular WebSocket, es una biblioteca cliente / servidor para Node.js que ayuda a administrar las cosas bajo el protocolo WebSocket. Como el protocolo WebSocket es compatible de forma nativa en la mayoría de los principales navegadores, algunos optan por trabajar con WS directamente sin usar socket.io en la parte superior.

**3. Sockjs**:

Sockjs es una biblioteca de JavaScript de navegador que proporciona un objeto similar a WebSocket con una API de Javascript de navegador múltiple que crea una comunicación de dominio cruzado, dúplex completo, dúplex completo y baja entre el navegador y el servidor web. Sigue la API de websocket HTML5 y solo cuando falla, recurre a los protocolos específicos del navegador. Sockjs-node es la contraparte de la biblioteca cliente.

**4. Node websocket:**

Esta herramienta websocket para Node.js es una implementación de JavaScript mayoritariamente pura de las versiones 8 y 13 del protocolo WebSocket para Node. WebSocket-Node incluye la funcionalidad de cliente y servidor, disponible a través de WebSocketClient y WebSocketServer.

**5. Faye-websocket-node**

Este proyecto es un cliente y servidor WebSocket que cumple con los estándares. Proporciona clases para crear servidores y clientes WebSocket en el nodo, pero no el servidor en sí ni ninguna abstracción adicional que no sea la API estándar de WebSocket (aparte de las conexiones eventSource).

**6. Websocket Promise**

Una biblioteca cliente WebSocket que proporciona una API basada en Promise para conectar, desconectar y enviar mensajes al servidor. Es básicamente un simple mecanismo de solicitud-respuesta que devuelve una promesa. Para trabajar con Node.js, debe combinarse con una biblioteca de servidores de nodos de terceros como websockets-node.

## **Compatibilidades:**

La API de WebSocket está disponible para el código JavaScript cuyo alcance DOM sea un objeto *Window* o cualquier objeto implementando *WorkerUtils*; si es así, puedes usarlos desde los Web Workers.

WebSocket es compatible con todos los navegadores modernos: Chrome 16 (incluido Chrome para Android), Firefox 11 (incluido Firefox para Android), Internet Explorer 10 (incl. Internet Explorer Mobile en Windows Phone 8), Safari 6, Opera 12.1 (incl. Opera Mobile) iOS 6.0, Blackberry 7.

## **Bibliografía:**

<https://developer.mozilla.org/es/docs/WebSockets-840092-dup>

<https://developer.mozilla.org/es/docs/WebSockets-840092-dup/Writing_WebSocket_client_applications>

<https://developer.mozilla.org/es/docs/WebSockets-840092-dup/Escribiendo_servidores_con_WebSocket>

<https://medium.com/bucaramangajs/real-time-websockets-25fa0cb520b4>

<https://crossbar.io/docs/Browser-Support/>

<https://es.wikipedia.org/wiki/WebSocket>

<https://blog.bitsrc.io/8-node-js-web-socket-libraries-for-2018-818e7e5b67cf>

<https://es.wikipedia.org/wiki/Cliente-servidor>

<https://es.wikipedia.org/wiki/Interfaz_de_entrada_com%C3%BAn>

<https://es.wikipedia.org/wiki/Middleware>

# **WSDL:**

## **Definición:**

WSDL es una notación XML para describir un servicio web. Una definición WSDL indica a un cliente cómo componer una solicitud de servicio web y describe la interfaz que proporciona el proveedor del servicio web.

Una definición WSDL se divide en varias secciones que especifican la interfaz lógica y los detalles físicos de un servicio web. Los detalles físicos incluyen información de puntos finales, como el número de puerto HTTP y la información de enlaces que especifica cómo se representa la carga útil SOAP y qué transporte se utiliza.

## **Analizar los modelos WSDL:**

RPC

Document

JAX

Lteral/encoded

SOAP

Port, EndPoint

Service

## **WS-I:**

Dentro de la arquitectura SOA para la implementación de Servicios Web, la interoperabilidad es tal vez el principio más importante. Como método de implementación de SOA, Web Services debe ofrecer importantes beneficios de interoperabilidad, y permitir la ejecución de servicios Web distribuidos en múltiples plataformas de software y arquitecturas de hardware.

La **Organización para la Interoperabilidad de Servicios Web** (*Web Services Interoperability Organization*) es un esfuerzo de la industria en este sentido. Su objetivo es fomentar y promover la Interoperabilidad de Servicios Web (*Web Services Interoperability* - **WS-I**) sobre cualquier plataforma, sobre aplicaciones, y sobre lenguajes de programación. Su intención es ser un integrador de estándares para ayudar al avance de los servicios web de una manera estructurada y coherente. La WS-I ha organizado los estándares que afectan a la interoperabilidad de los servicios web en una *pila* basada en funcionalidades.

## **¿Cómo afecta la generación del stub que enviará los mensajes SOAP?:**

## **Bibliografía:**

<https://es.wikipedia.org/wiki/WS-I>

<https://www.ibm.com/support/knowledgecenter/es/SSMKHH_10.0.0/com.ibm.etools.mft.doc/ac34640_.htm>

<http://di002.edv.uniovi.es/~falvarez/WSDL.pdf>