# Índice

[Introducción](#h.btoyc59yq8y0)

[Modelos de comunicación](#h.1uemwlnm2jtf)

[Http estandar:](#h.q6a1s3m7dznw)

[AJAX Polling:](#h.5ar4i7wkmk39)

[AJAX Long-Polling:](#h.547hzm9c0cml)

[HTML5 Server Sent Events (SSE) / EventSource:](#h.j4tid86wrykd)

[WebSockets](#h.w1coggokxfrk)

[Modelo de comunicación por WebSockets](#h.qx93cvshqlan)

[Funcionamiento del protocolo WebSocket](#h.1xbowvkwamjn)

[Establecimiento de conexión (handshake)](#h.sp7rlfogvq4y)

[Comunicación con el servidor](#h.mi0ijwoxub47)

[Referencias bibliográficas](#h.kfn8fqrjm8n4)

# 

# 

# Introducción

La arquitectura web básica se compone generalmente de dos actores, por un lado el cliente y por el otro, un servidor, que interactúan mediante peticiones. El servidor generalmente recibe las peticiones de los clientes, las procesa y envía posteriormente los resultados a través de la conexión que existe entre ellos.

En sus comienzos el modelo Cliente-Servidor servía las páginas de una manera sincrónica, lo que significa que por cada petición del cliente a través de un request HTTP, el cliente quedaba bloqueado hasta que el servidor daba una respuesta.

A fines de la década del 90 se realizaron los primeros intentos de dinamizar la web utilizando iframes ocultos, lo que permitía efectuar consultas asincrónicamente en un iframe y luego, mediante javascript, procesar la respuesta de la misma.

A mediados del 2000 los navegadores comenzaron a implementar el objeto XMLHttpRequest (XHR) que permite conexiones asíncronas con el servidor, utilizando javascript, lo que dió paso a una nueva técnica denominada **AJAX**.

Sin embargo, muchas veces es requerido que el servidor informe al cliente de eventos sin que el cliente haya realizado un request para pedir información. Para resolver este problema, fue incorporada la tecnología **COMET** (también denominada Push), donde se combinan una serie de un modelos y técnicas de comunicacion (como AJAX con long polling, iframes ocultos, etc) que permiten emular una transacción originada en el servidor. Con esta tecnología, el cliente realiza un primer request y luego queda esperando, en una conexión abierta, que el servidor vaya respondiendo en la medida que lo requiera .

El problema de las tecnologías descriptas es que ninguna representa una solución aceptable en aplicaciones web de baja latencia, como juegos en línea, chats o redes sociales, donde el tiempo de respuesta, y la optimización del canal de comunicación son de vital importancia.

Por otro lado, en los casos donde se realiza un request asíncrono y el servidor no tiene información para dar, se genera un gasto innecesario de tráfico en la red porque la petición del cliente al servidor son datos que ocupan memoria y ancho de banda.

Otro gran problema de estas técnicas es que solo permiten un esquema de comunicación Half-Duplex entre el cliente y el servidor, ya que los dos pueden enviar y recibir mensajes pero solo uno a la vez puede utilizar el canal de comunicación.

# 

# Modelos de comunicación

## Http estandar:HTTP

1. El cliente solicita una página web al servidor (request).
2. El servidor procesa la respuesta.
3. El servidor envía la respuesta al cliente (response).

## AJAX Polling:AJAX Polling

1. El cliente solicita una página web utilizando http estándar.
2. El servidor procesa la respuesta y la envía al cliente.
3. La página solicitada ejecuta un script (javascript), donde se realizan requests al servidor a intervalos regulares (0,5 segundos, por ejemplo).
4. El servidor procesa cada respuesta (si la hay), y la envía.

## AJAX Long-Polling:AJAX Long-Polling

1. El cliente solicita una página web utilizando http estándar.
2. El servidor procesa la respuesta y envía la respuesta al cliente.
3. La página solicitada ejecuta un script (javascript), donde se realiza un request al servidor.
4. El servidor no responde inmediatamente, sino que espera a que haya nueva información disponible.
5. Cuando hay nueva información disponible, el servidor procesa el response y lo envía.
6. El cliente recibe la nueva información e inmediatamente envía otro request al servidor, reiniciando el proceso.

## HTML5 Server Sent Events (SSE) / EventSource:HTML5 SSE

1. El cliente solicita una página web utilizando http estándar.
2. El servidor procesa la respuesta y envía la respuesta al cliente.
3. La página solicitada ejecuta un script (javascript), donde se abre una conexión con el servidor
4. Cuando hay nueva información disponible, el servidor procesa el response y lo envía, sin cerrar la conexión.

# 

# 

# WebSockets

Websockets es una tecnología que llega para resolver los problemas de comunicación que plantean los esquemas descriptos. De manera simple, Websockets permite comunicar el cliente y el servidor a través de un canal Full Duplex bidireccional y sin tener que hacer polling por parte del cliente.

## Modelo de comunicación por WebSocketsHTML5 WebSockets

1. El cliente solicita una página web utilizando http estándar.
2. El servidor procesa la respuesta y envía la respuesta al cliente.
3. La página solicitada ejecuta un script (javascript), donde se abre una conexión con el servidor
4. El servidor y cliente pueden enviarse información (en ambas direcciones) cuando haya nueva información disponible..

La especificación WebSocket define un API que establece conexiones "socket" entre un navegador web y un servidor. Dicho con otras palabras: existe una conexión persistente entre el cliente y el servidor, y ambas partes pueden empezar a enviar datos en cualquier momento.

El protocolo WebSocket pretende superar la limitación estructural del protocolo HTTP. Como las conexiones TCP ordinarias sobre puertos diferentes al 80 son habitualmente bloqueadas, el uso de esta tecnología proporciona 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).

Actualmente, la API de WebSocket está siendo normalizada por el W3C, y el protocolo WebSocket, a su vez, está siendo normalizado por el IETF.

## Funcionamiento del protocolo WebSocket

### Establecimiento de conexión (handshake)

El protocolo WebSocket para la comunicación bidireccional requiere que tanto la aplicación cliente como servidor conozcan los detalles del protocolo. Esto quiere decir que requiere de una página web que cumpla con WebSocket y que se comunique con un extremo que cumpla con WebSocket.

La interacción de WebSocket comienza con un protocolo de enlace en el que las dos partes (explorador y servidor) confirman mutuamente su intención de comunicarse a través de una conexión persistente. Cuando se cierra la conexión, ambos extremos intercambian un marco de cierre para cerrar la conexión en forma limpia. El enlace inicial consiste en una simple solicitud HTTP que el cliente envía al servidor web. La solicitud es una instrucción HTTP GET configurada como una solicitud de actualización:

|  |
| --- |
| GET /chat HTTP/1.1  Host: server.example.com  Upgrade: websocket  Connection: Upgrade  Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==  Origin: http://example.com |

En HTTP, una solicitud del cliente con un encabezado de actualización indica que el cliente tiene la intención de solicitarle un cambio de protocolo al servidor. Con el protocolo WebSocket, la solicitud de actualización al servidor contiene una clave única que el servidor devolverá modificada, como prueba de que aceptó la solicitud de actualización. Esta demostración práctica muestra que el servidor comprende el protocolo WebSocket. Aquí vemos un ejemplo de una respuesta a una solicitud de enlace:

|  |
| --- |
| HTTP/1.1 101 WebSocket Protocol Handshake  Upgrade: websocket  Connection: Upgrade  Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo= |

El código de estado satisfactorio siempre es 101 y cualquier otro código de estado se interpreta como una negativa a cambiar al protocolo WebSocket. El servidor concatena la clave recibida con una cadena GUID fija y calcula un valor hash a partir de la cadena resultante. Luego, lo codifica en Base64 y lo devuelve al cliente a través del encabezado Sec-WebSocket-Accept.

Después del protocolo de enlace, el cliente y el servidor pueden enviar mensajes libremente a través del protocolo WebSocket. La carga comienza con un código de operación que señala la operación que se está llevando a cabo. Uno de estos códigos de operación (específicamente 0x8) indica una solicitud para cerrar la sesión.

Los mensajes de WebSocket se transmiten de manera asincrónica, por lo que las solicitudes enviadas no recibirán necesariamente una respuesta inmediata, como en HTTP. Con el protocolo WebSocket, conviene pensar en términos de mensajes generales que van del cliente al servidor o viceversa y olvidar los patrones clásicos de solicitud y respuesta de HTTP.

### Comunicación con el servidor

Para abrir una conexión WebSocket, solo se debe ejecutar el constructor WebSocket:

|  |
| --- |
| var myWebSocket = new WebSocket("ws://[www.websocket.org](http://www.websocket.org)"); |

Se puede utilizar el prefijo de protocolo wss, si se desea usar una conexión de socket segura (de la misma forma que se utiliza https: para las conexiones HTTP seguras).

Una vez creado el Websocket lo siguiente que tenemos que hacer es definir las funciones de callback que controlarán el intercambio de información con el Websocket:

|  |
| --- |
| *// Callback al establecer la conexión*  connection.onopen = function () {  connection.send('Ping'); *//Se envía el mensaje "Ping" al servidor.*  };  *// Callback de error*  connection.onerror = function (error) {  console.log('Error en WebSocket ' + error);  };  *//Callback cuando se recibe un nuevo mensaje del servidor*  connection.onmessage = function (e) {  console.log('Mensaje del servidor: ' + e.data);  }; |

# Referencias bibliográficas

* <http://es.wikipedia.org/wiki/WebSocket>
* <http://stackoverflow.com/questions/11077857/what-are-long-polling-websockets-server-sent-events-sse-and-comet>
* <http://www.html5rocks.com/es/tutorials/websockets/basics/>
* <http://www.davesite.com/html5-code-tutorials/html5_interactive_intro_html_5/websockets/>
* <http://www.pointsoftware.ch/de/websocket-versus-comet-real-time-web-applications/>
* <http://superwebsocket.codeplex.com/>
* <http://msdn.microsoft.com/es-es/magazine/hh975342.aspx>
* <https://sites.google.com/site/gabineteutn/investigacion-y-desarrollo/html5/tutoriales/introduccion-a-websocket>
* <http://msdn.microsoft.com/es-es/library/ie/hh673567(v=vs.85).aspx>