Websockets.

Índice.

1 Introducción.

1. Introducción.

Características:

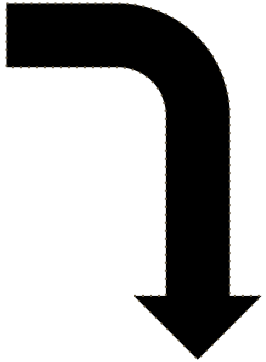
- Comunicación bidireccional.
- Full-dúplex.
- Protocolo TCP.

- Estandarizado por la **W3C** (World Wide Web Consortium).

Requisitos: Websockets tanto en el lado del cliente como en el lado del servidor.

1. Introducción.

HTML



Websocket.

1. Para realizar una conexión WebSocket se intercambia una serie de información entre cliente y servidor bajo el protocolo HTTP.
2. Si no surgen fallos se actualiza de HTTP a WebSocket, usando la conexión TCP ya establecida.

```
wss://
```

2. Problema baja latencia.

2. Problema baja latencia.

La latencia en las comunicaciones es otro de los beneficios de utilizar WebSockets.

El socket está siempre abierto y en escucha, por lo que los datos se envían inmediatamente desde el servidor al navegador, reduciendo considerablemente el tiempo de espera.

3. Utilidades.

3. Utilidades.

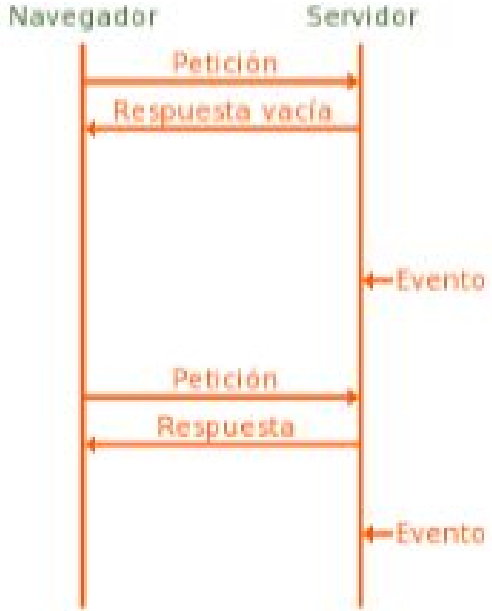
Básicamente, **aplicaciones de tiempo real.**

Antiguamente, se usaban las siguientes técnicas.

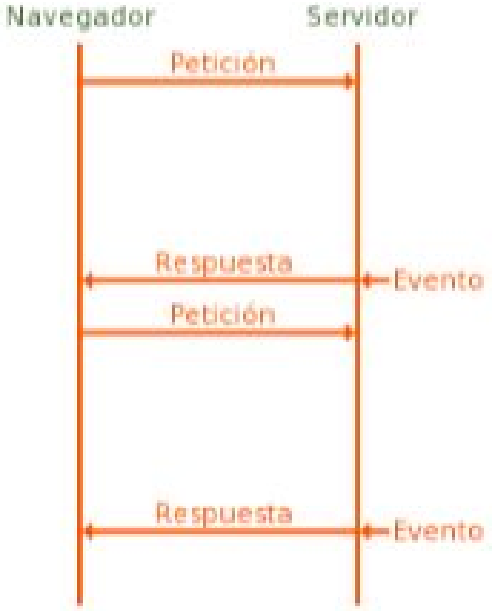
1. AJAX Polling.
2. Long Polling.

Ajax vs. Comet

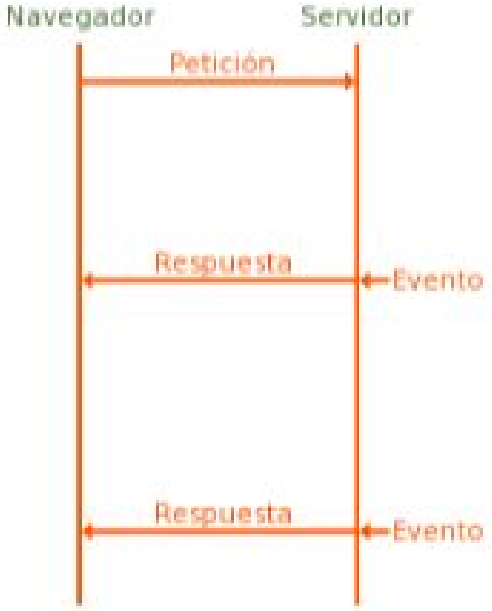
Ajax



Comet
(con long-polling)



Comet
(con websockets)



4. Uso actual.

4. Uso actual.

- Trabajos colaborativos.
- Juegos multijugador.
- Chat.
- Herramienta de monitorización.

5. Lenguajes que soporta.

5. Lenguajes que soporta.

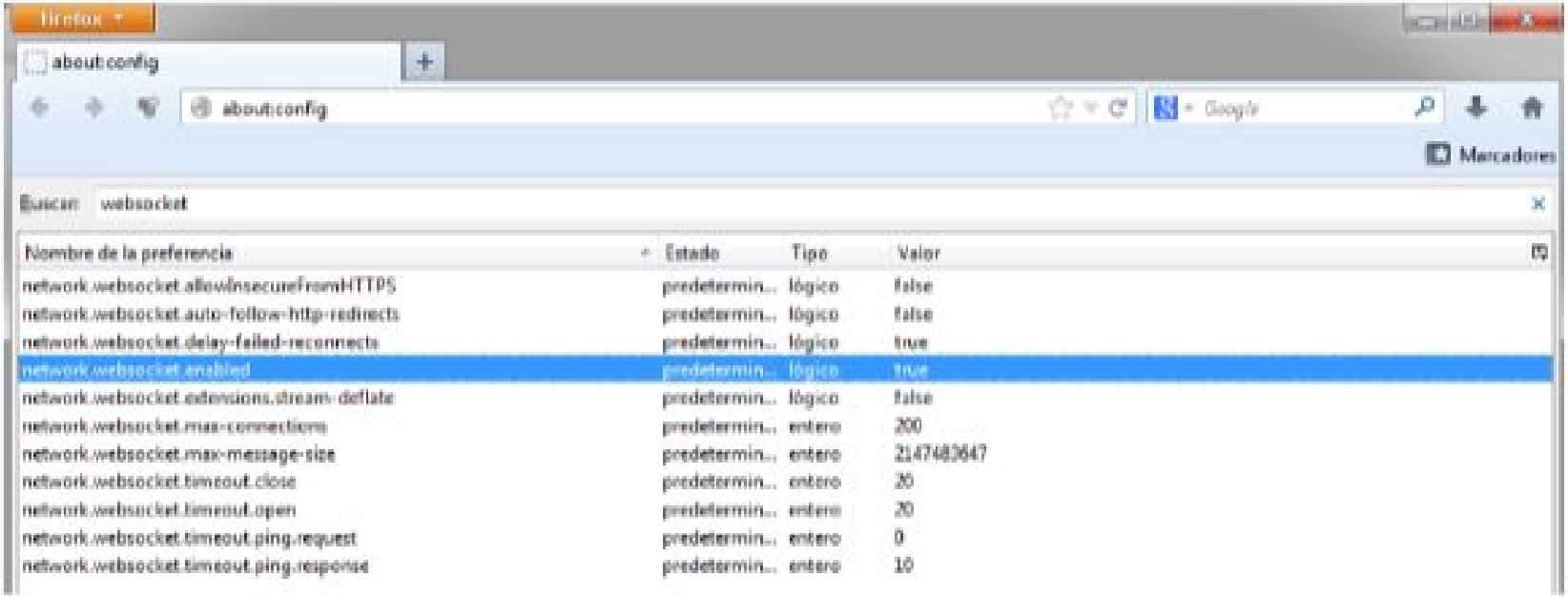
- ★ Node .js.
- ★ Java.
- ★ Ruby.
- ★ Python.
- ★ Erlang.
- ★ C++.
- ★ .NET

6. Comparativa de compatibilidades y activación en navegadores.

6. Comparativa de compatibilidades y activación en navegadores.

http://caniuse.com/#feat=websockets

6. Activación de navegadores.



7. Caso real.

8. Cómo crear un chat usando WebSocket.


CLIENTE
HTML + CSS
JAVASCRIPT
API WS
SERVIDOR
PHP

DISEÑO DEL CLIENTE

index.html

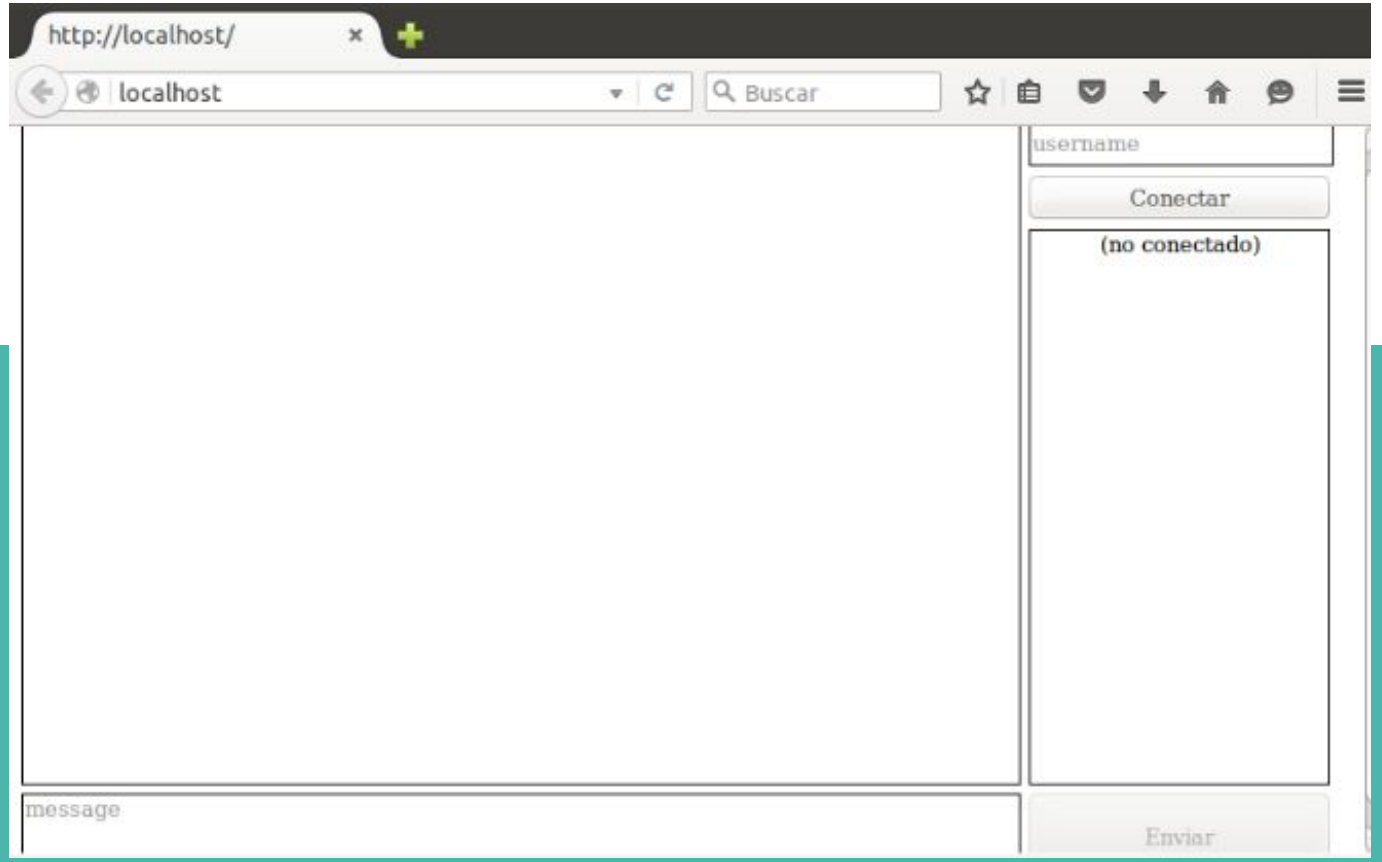
```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" type="text/css" href="main.css" />
    <script type="text/javascript" src="main.js"></script>
</head>

<body>
    <div class="box1">
            <textarea id="chatLines" readonly="readonly"></textarea><br />
            <textarea id="chatInput" placeholder="message"></textarea>
    </div>

    <div class="box2">
            <input id="usernameInput" type="text" placeholder="username" /><br />
            <input id="connectButton" type="button" value="Conectar" onclick="cb.onClickConnect()" /><br />
            <ul id="usersList"><li style="text-align:center;">(no conectado)</li></ul>
            <input id="sendButton" type="button" value="Enviar" onclick="cb.onClickSend()" />
    </div>

</body>

</html>

```

DISEÑO DEL CLIENTE

index.html × main.css ×

```
1  * {
2      font-family: Verdana;
3      font-size: 13px;
4  }
5
6  .box1 {
7      float: left;
8  }
9  .box2 {
10     float: left;
11     margin-left: 5px;
12 }
13
14 #chatLines {
15     width: 600px;
16     height: 400px;
17     margin: 0px;
18     margin-bottom: 5px;
19     border: solid 1px #000000;
20     resize: none;
21 }
22 #chatInput {
23     width: 600px;
24     height: 50px;
25     margin: 0px;
26     border: solid 1px #000000;
27     resize: vertical;
28 }
29
30 #usernameInput {
31     width: 180px;
32     height: 24px;
33     margin-bottom: 5px;
34     border: solid 1px #000000;
35 }
36 #connectButton {
37     width: 184px;
38     height: 28px;
39     margin-left: -1px;
40     margin-bottom: 5px;
41 }
42 #usersList {
```

DISEÑO DEL CLIENTE



API WEBSOCKET

api.php

```php
<?php

/*
    PHP WebSocket Server 0.2
     - http://code.google.com/p/php-websocket-server/
     - http://code.google.com/p/php-websocket-server/wiki/Scripting

    WebSocket Protocol 07
     - http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-07
     - Supported by Firefox 6 (30/08/2011)

    Whilst a big effort is made to follow the protocol documentation, the current script version may unknowingly differ.
    Please report any bugs you may find, all feedback and questions are welcome!
*/


// settings

// maximum amount of clients that can be connected at one time
define('WS_MAX_CLIENTS', 100);

// maximum amount of clients that can be connected at one time on the same IP v4 address
define('WS_MAX_CLIENTS_PER_IP', 15);

// amount of seconds a client has to send data to the server, before a ping request is sent to the client,
// if the client has not completed the opening handshake, the ping request is skipped and the client connection is closed
define('WS_TIMEOUT_RECV', 10);

// amount of seconds a client has to reply to a ping request, before the client connection is closed
define('WS_TIMEOUT_PONG', 5);

// the maximum length, in bytes, of a frame's payload data (a message consists of 1 or more frames), this is also
internally limited to 2,147,479,538
define('WS_MAX_FRAME_PAYLOAD_RECV', 100000);

// the maximum length, in bytes, of a message's payload data, this is also internally limited to 2,147,483,647
define('WS_MAX_MESSAGE_PAYLOAD_RECV', 500000);




// internal
```

CONEXIÓN DE UN USUARIO

```
onClickConnect: function() {

    if (chat.isConnected()) {
        chat.disconnect();
        return;
    }


    var Username = document.getElementById('usernameInput').value;

    // validamos el usuario
    if (Username == '') {
        chat.log('Debes poner un nombre de usuario.');
    }
    else if (Username.indexOf(' ') != -1) {
        chat.log('El nombre de usuario no puede contener espacios.');
    }
    else if (Username.length > chat.usernameMaxLength) {
        chat.log('El nombre de usuario no puede tener mas de '+chat.usernameMaxLength+' caracteres.');
    }
    else {
        // conectamos con el servidor
        chat.setElementsDisabled(true);
        chat.log('Conectando..');
        chat.connect();
    }
},
```

```
// realiza la conexion al servidor, creamos objeto y habilitar eventos.
connect: function() {
    var Socket = new WebSocket('ws://'+chat.serverHost+':'+chat.serverPort);
    chat.setSocketEvents(Socket);
    chat.socket = Socket;
},
```

```
// metodo que inserta un texto dentro del objeto chatlines
log: function(Text) {
    // inserta texto en una nueva linea si chatline tiene texto se hace \n
    var ChatLinesElement = document.getElementById('chatLines');
    ChatLinesElement.value += ((ChatLinesElement.value != '') ? '\n' : '') + Text;

    // visualizar la propiedad scroll
    ChatLinesElement.scrollTop = 100000;
},
```

ENVÍO DE MENSAJES

```
//funcion cuando un usuario envia un mensaje
onClickSend: function() {


    var ChatInputElement = document.getElementById('chatInput');
    var Text = ChatInputElement.value;

    // comprobamos conexion y el texto
    if (!chat.isConnected()) {
        chat.log('No conectado.');
    }
    else if (Text == '') {
        chat.log('Debes escribir un texto.');
    }
    else {
        // se envia el tenxto al servidor con la cabecera TEXT
        chat.socket.send('TEXT '+Text);

        // limpiar el campo chatinput
        ChatInputElement.value = '';
    }
},
```

```
function wsOnMessage($clientID, $message, $messageLength, $binary) {
    // comprueba la longitud
    if ($messageLength == 0) {
        wsClose($clientID);  // llama a api para cerrar la comunicacion
        return;
    }

    // si no esta vacio crea un array para capturar las cabeceras
    $message = explode(' ', $message);
    $command = array_shift($message);

    // comprueba la cabecera
    if ($command == 'TEXT') {
        // un usuario ha enviado un texto al servidor, comprueba el usuario sea valido

        if (!isUser($clientID)) {
            wsClose($clientID);
            return;
        }

        // coloca el mensaje en la cadena string (implode junta los elementos del string)
        $text = implode(' ', $message);

        if ($text == '') {
            // si esta vacio
            wsSend($clientID, 'SERVER Mensaje vacio.');
            return;
        }

        //recojo en nombre de usuario y mediante la funcion sendchat (api) envio a todos

        $username = getUsername($clientID);
        sendChat($username, $text);
    }
```

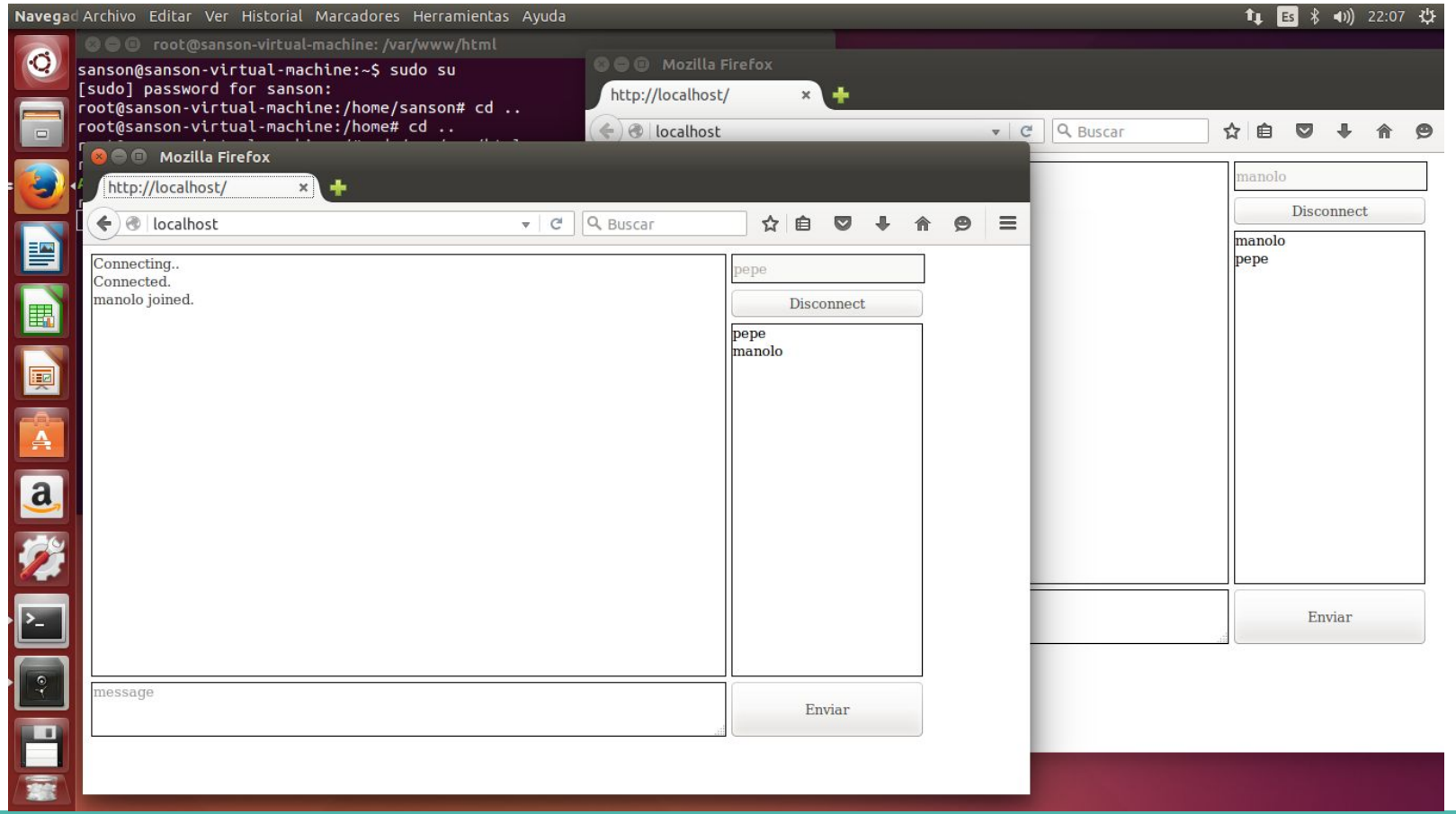
DESCONEXIÓN DE USUARIO

```
//desconectar
disconnect: function() {
    chat.socket.send('QUIT');
    chat.socket.close();
},
```

```
elseif ($command == 'QUIT') {
    // un usuario abandona el chat

    if (!isUser($clientID)) {  // comprueba por seguridad que no se pueda enviar quit sin ser
    usuario validado
        wsClose($clientID);
        return;
    }

    // esta en la lista y hace quit elimina usuario
    removeUser($clientID);
}
```

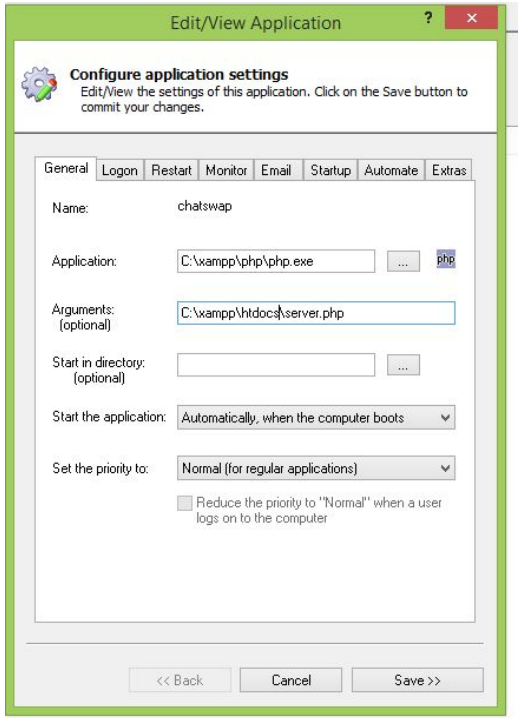
```
function removeUser($clientID) {
    global $users;

    // recoge el nombre usuario por su id
    $username = getUsername($clientID);

    // lo borramos de la lista de usuarios users
    unset($users[$clientID]);

    // envia a todos que ese usuario ha sido desconectado
    foreach ($users as $clientID2 => $username2) {
        wsSend($clientID2, 'ONQUIT '.$username);
    }
}
```

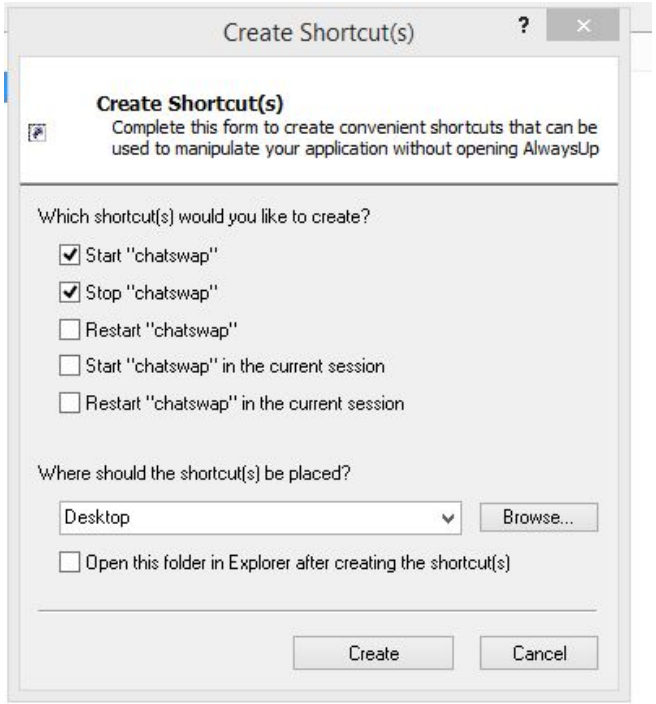
DEMO


Navegad Archivo Editar Ver Historial Marcadores Herramientas Ayuda
22:07
root@sanson-virtual-machine: /var/www/html
sanson@sanson-virtual-machine:~$ sudo su
[sudo] password for sanson:
root@sanson-virtual-machine:/home/sanson# cd ..
root@sanson-virtual-machine:/home# cd ..
Mozilla Firefox
http://localhost/
localhost
Buscar
Mozilla Firefox
http://localhost/
localhost
Buscar
Connecting..
Connected.
manolo joined.
pepe
Disconnect
pepe
manolo
message
Enviar
manolo
Disconnect
manolo
pepe
Enviar

WS como Soporte Tecnico

Windows-AlwaysUp-WebSockets





¿Es seguro?

Existen mecanismos más seguros y potentes para efectuar este tipo de comunicaciones.

Para PHP por ejemplo **WebRatcket**

http://socketo.me/

9. Ventajas y desventajas.

Ventajas & Desventajas.

- Posibilidad de comunicaciones bidireccionales en tiempo real.
- Reduce la saturación existente con HTTP.
- Funcionamiento sencillo.
- Mismos puertos que HTTP.

- Gestionar las numerosas conexiones que se mantienen abiertas mientras ambas partes interactúan.
- Problema con TCP mx. conexiones (64000)
- Necesita memoria del servidor para mantener las conexiones abiertas.

10. Conclusiones.

Conclusiones.

- Aplicaciones con comunicaciones bidireccionales.
- Multiplataforma y multilenguaje.
- No es necesario implementar las comunicaciones.
- Muy buena opción para aplicaciones que necesiten actualizarse en tiempo real (chats, juegos multijugador en línea o retransmisiones interactivas en directo).
- Hay que tener en cuenta la memoria del servidor para mantener las conexiones simultáneas.

Referencias.

http://www.loxone.com/eses/servicio/documentacion/visualizacion/compatibilidad.html

https://azure.microsoft.com/es-es/blog/introduction-to-websockets-on-windows-azure-web-sites/

https://www.w3.org/TR/websockets/

http://caniuse.com/#feat=websockets

http://www.html5rocks.com/en/tutorials/websockets/basics/

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

https://code.msdn.microsoft.com/windowsapps/Connecting-with-WebSockets-643b10ab/

http://www.adictosaltrabajo.com/tutoriales/web-sockets-java-tomcat/

https://developer.mozilla.org/es/docs/WebSockets

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

Referencias.

http://www.arkaitzgarro.com/html5/capitulo-13.html

http://www.arquitecturajava.com/java-websockets/

https://www.nuget.org/packages/WebSocket.Portable.Core/

http://lineadecodigo.com/html5/crear-un-websocket/

http://www.htmlgoodies.com/html5/other/create-a-bi-directional-connection-to-a-php-server-using-html5-websockets.

html#fbid=FDhH9_nbENA

https://www.sanwebe.com/2013/05/chat-using-websocket-php-socket

http://www.sitepoint.com/how-to-quickly-build-a-chat-app-with-ratchet/

http://socketo.me/docs/websocket