

Inicio (<https://recursospython.com/>)

Códigos de fuente (<https://www.recursospython.com/category/codigos-de-fuente/>)

Guías y manuales (<https://www.recursospython.com/category/guias-y-manuales/>)

Foro (<https://foro.recursospython.com/>)

Micro (<https://micro.recursospython.com/>)

Tutorial (<https://tutorial.recursospython.com/>)


Newsletter (<https://recursospython.com/newsletter/>)

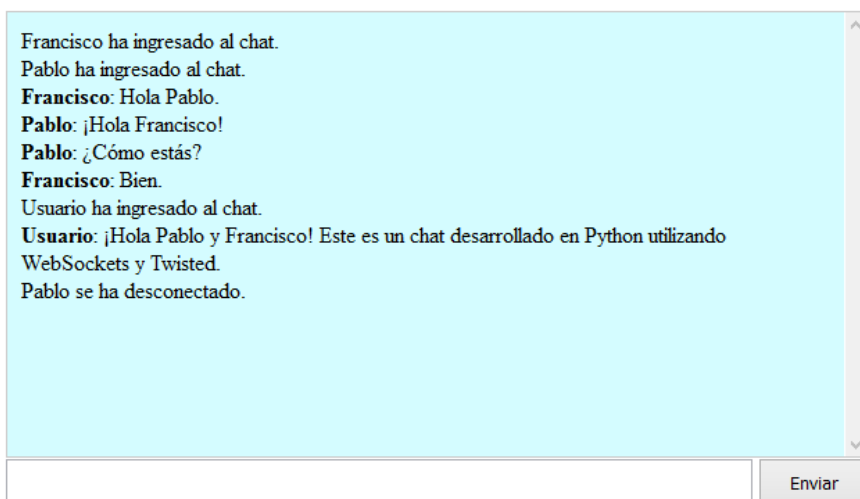
Consultoría (<https://recursospython.com/consultoria/>)

Contacto (<https://recursospython.com/contacto/>)

Donar  (<https://recursospython.com/donar/>)

Chat vía web con WebSockets y Twisted

julio 12, 2016 (<https://recursospython.com/codigos-de-fuente/chat-via-web-websockets-twisted/>) by [Recursos Python](https://recursospython.com/author/admin/) (<https://recursospython.com/author/admin/>) 
(<https://recursospython.com/codigos-de-fuente/chat-via-web-websockets-twisted/#comments>) 8 comentarios
(<https://recursospython.com/codigos-de-fuente/chat-via-web-websockets-twisted/#comments>)



(<https://www.recursospython.com/wp-content/uploads/2016/07/preview.png>)



Últimas entradas

[Reproducir inyección](#)

[SQL en sqlite3 y](#)

[PyMySQL](#)

(<https://recursospython.com/guias-y-manuales/reproducir-inyeccion-sql-en-sqlite3-y-pymysql/>)

[Bloc de notas simple con](#)

[Tk \(tkinter\)](#)

(<https://recursospython.com/codigos-de-fuente/bloc-de-notas-simple-con-tkinter/>)

[Examinar archivo o](#)

[carpeta en Tk \(tkinter\)](#)

(<https://recursospython.com/guias-y-manuales/examinar-archivo-o-carpeta-en-tk-tkinter/>)

WebSocket es una tecnología que permite realizar conexiones y transmitir información cliente / servidor de forma similar a un socket convencional, pero que es soportada por los principales navegadores web. Se trata de una herramienta relativamente reciente, por ende no está disponible en versiones antiguas de dichos programas. Cuenta con una API estandarizada por el [World Wide Web Consortium \(W3C\)](https://www.w3.org/TR/2011/WD-websockets-20110929/) (<https://www.w3.org/TR/2011/WD-websockets-20110929/>), a la cual se accede vía JavaScript. Los WebSockets resultan una herramienta de mucho potencial, que abre un mundo nuevo de posibilidades para las aplicaciones web.

Aquí en Recursos Python somos fanáticos de Twisted. Por ende, mientras que la parte del cliente es manejada por el navegador vía JavaScript, estaremos desarrollando el servidor utilizando dicho framework. Ya que Twisted no soporta de forma nativa el protocolo de WebSocket, utilizamos una [pequeña librería](https://pypi.python.org/pypi/txWS) (<https://pypi.python.org/pypi/txWS>) para añadir esta característica.

Para introducir al lector a los WebSockets de una forma agradable, a lo largo del artículo estaremos desarrollando, paso a paso, un simple chat web. Preferentemente se requieren conocimientos básicos de HTML, JavaScript y [Twisted](https://www.recursospython.com/guias-y-manuales/introduccion-a-twisted/) (<https://www.recursospython.com/guias-y-manuales/introduccion-a-twisted/>).

Código completo y descarga al final del artículo.

Cliente

Comenzaremos por crear el archivo `chat.js`, que tendrá todo el código necesario para comunicarse con el servidor y desplegar los mensajes en el navegador. Implementamos dos variables globales; una representa el WebSocket y otra almacena el nombre del usuario en la conversación.

```
1. var ws;  
2. var username;
```

Dejamos por un momento el código JavaScript para crear el archivo HTML principal, `wscclient.html`.

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
2. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
3. <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es"  
   lang="es">  
4.   <head>  
5.     <title>Chat</title>  
6.     <meta http-equiv="Content-Type" content="text/html;  
   charset=utf-8" />  
7.     <script src="chat.js"></script>  
8.   </head>  
9.   <body>
```

[Múltiples configuraciones \(desarrollo/producción\) en Django](#) (<https://recursospython.com/guias-y-manuales/multiples-configuraciones-desarrollo-produccion-en-django/>)

[Buscar el archivo de mayor tamaño en una ruta](#) (<https://recursospython.com/comentarios-de-fuente/buscar-el-archivo-de-mayor-tamano-en-una-ruta/>)

Comentarios recientes

Recursos Python en [Generar código QR](#) (<https://recursospython.com/guias-y-manuales/generar-codigo-qr/#comment-2586>)

Joaquín en [Generar código QR](#) (<https://recursospython.com/guias-y-manuales/generar-codigo-qr/#comment-2584>)

Recursos Python en [pickle – Serialización de objetos](#) (<https://recursospython.com/guias-y-manuales/pickle-serializacion-de-objetos/#comment-2435>)

Recursos Python en [Lista desplegable \(Combobox\)](#)

```
10.     </body>
11. </html>
```

En el código anterior simplemente creamos una estructura básica HTML, establecemos un título (línea número 5) e importamos el archivo `chat.js` (7).

Ahora introducimos la primera parte del chat, el momento de elegir un nombre para la conversación e ingresar (dentro de las etiquetas `body`).

```
1.     <div id="login">
2.         <label>Nombre: </label> <input autocomplete="off"
type="text" id="username" value="Usuario" />
3.         <button onclick="loadChat();">Ingresar</button>
4.     </div>
```

El botón de ingresar llamará a la función `loadChat`, que será la encargada de guardar el nombre de usuario y establecer la conexión. La definimos de la siguiente forma.

```
1. // chat.js
2. function loadChat()
3. {
4.     username = document.getElementById("username").value;
5.     if (!username)
6.         return;
7.
8.     document.getElementById("login").hidden = true;
```

Hasta aquí obtenemos el nombre introducido por el usuario, chequeamos que no esté vacío, y ocultamos el recuadro. Retomamos esta función más adelante.

Regresamos a nuestro archivo HTML y, debajo de lo último que hemos añadido, colocamos el siguiente recuadro.

```
1.     <div id="chat" hidden>
2.         <div id="messages" style="width: 600px; height:
300px; overflow-y: scroll; background-color: #D4FCFF;
border: 1px solid #CACACA; padding: 10px;">
3.             Aún no te has conectado al chat.
4.         </div>
5.         <input type="text" style="width: 530px; height: 30px;
padding-left: 5px; margin: 0px;" id="message"
6.             onkeyup="onKeyUp(event);" />
7.         <button style="width:81px; height: 36px; padding:
0px; margin: 0px;" onclick="sendMessage();">Enviar</button>
8.         <br /><br />
9.         <button onclick="closeChat();">Salir</button>
10.     </div>
```

[en Tcl/Tk \(tkinter\).
\(https://recursospython.com/gui-y-manuales/lista-desplegable-combobox-en-tkinter/#comment-2434\)](https://recursospython.com/gui-y-manuales/lista-desplegable-combobox-en-tkinter/#comment-2434)
Herná en [Lista desplegable \(Combobox\) en Tcl/Tk \(tkinter\).
\(https://recursospython.com/gui-y-manuales/lista-desplegable-combobox-en-tkinter/#comment-2424\)](https://recursospython.com/gui-y-manuales/lista-desplegable-combobox-en-tkinter/#comment-2424)

Hasta aquí hemos introducido un recuadro en donde aparecerán los mensajes de la conversación (`id="messages"`), una entrada de texto para escribir los mensajes (`id="message"`), un botón para enviarlos (línea número 7) y otro para salir (9). Inicialmente se encuentra oculto, por lo que retomamos la función `loadChat` que dejamos a medias e introducimos lo siguiente.

```
1.      document.getElementById("chat").hidden = false;
2.
3.      messages = document.getElementById("messages");
4.      messages.innerHTML = ""; // Remover el contenido
      anterior.
```

De esta forma, una vez ingresado el nombre de usuario se mostrará el recuadro de la conversación.

Llega el momento de crear el `WebSocket` y establecer la conexión (debajo del código anterior). Nuestro servidor Twisted estará escuchando en el puerto número 9998.

```
1.      ws = new WebSocket("ws://localhost:9998");
```

Y creamos tres funciones para manejar los distintos eventos del `WebSocket`.

```
1.      // Al crearse la conexión.
2.      ws.onopen = function()
3.      {
4.          ws.send(username + " ha ingresado al chat.");
5.      };
6.
7.      // Al cerrarse la conexión.
8.      ws.onclose = function()
9.      {
10.         chat.innerHTML = "Se ha perdido la conexión."
11.      };
12.
13.     // Al recibir un mensaje.
14.     ws.onmessage = function(evt)
15.     {
16.         // Añadir el mensaje al recuadro.
17.         messages.innerHTML += evt.data + "<br />";
18.         // Desplegar la barra vertical hacia abajo.
19.         messages.scrollTop = messages.scrollHeight;
20.     };
21. }
```

La función completa:

```
1.      function loadChat()
2.      {
3.          username = document.getElementById("username").value;
4.          if (!username)
5.              return;
6.          ▲
```

```

7.     document.getElementById("login").hidden = true;
8.     document.getElementById("chat").hidden = false;
9.
10.    messages = document.getElementById("messages");
11.    messages.innerHTML = "";
12.
13.    ws = new WebSocket("ws://localhost:9998");
14.
15.    ws.onopen = function()
16.    {
17.        ws.send(username + " ha ingresado al chat.");
18.    };
19.
20.    ws.onclose = function()
21.    {
22.        chat.innerHTML = "Se ha perdido la conexión."
23.    };
24.
25.    ws.onmessage = function(evt)
26.    {
27.        messages.innerHTML += evt.data + "<br />";
28.        messages.scrollTop = messages.scrollHeight;
29.    };
30. }

```

Continuamos implementando la función `sendMessage`, encargada de enviar un mensaje al servidor a través del WebSocket.

```

1.  function sendMessage()
2.  {
3.      // Obtener el mensaje ingresado por el usuario.
4.      message = document.getElementById("message");
5.      if (message.value)
6.      {
7.          // Enviar al servidor.
8.          ws.send("<strong>" + username + "</strong>: " +
message.value);
9.          // Borrar el contenido de la caja de texto.
10.         message.value = "";
11.     }
12.     // Enviar el foco nuevamente a la caja de texto.
13.     message.focus();
14. }

```

Y para permitir al usuario enviar mensajes presionando la tecla Enter.

```

1.  function onKeyUp(event)
2.  {
3.      if (event.keyCode == 13)
4.          sendMessage();
5.  }

```

Por último, la opción para salir de la conversación.

```

1.  function closeChat()
2.  {
3.      ws.send(username + " se ha desconectado.");
4.      ws.close();

```

```
5.     }
```

Ahora bien, como comentamos anteriormente, al tratarse de una tecnología relativamente nueva debemos chequear que el navegador soporte WebSockets.

```
1.  function checkSupport()
2.  {
3.      if (!("WebSocket" in window))
4.      {
5.          document.getElementById("login").innerHTML = "Este
navegador no soporta WebSockets.";
6.      }
7.  }
```

Esta función debe ser llamada una vez cargada la página, por lo que la colocamos justo antes de cerrar la etiqueta `body`.

```
1.      <script>checkSupport();</script>
2.      </body>
```

Con esto finalizamos la parte del cliente.

Servidor

El código para el servidor es bastante pequeño. Utilizamos Twisted (<http://twistedmatrix.com>) y el módulo txWS (<https://pypi.python.org/pypi/txWS>) que le añade soporte para WebSockets.

Creamos el archivo `wsserver.py` y comenzamos por importar todo lo necesario.

```
1.  from twisted.internet import protocol, reactor, endpoints
2.  from txws import WebSocketFactory
```

Luego, creamos el protocolo.

```
1.  class ClientProtocol(protocol.Protocol):
2.
3.      def __init__(self, factory):
4.          self.factory = factory
5.
6.      def dataReceived(self, data):
7.          """Enviar el mensaje a todos los clientes."""
8.          self.factory.sendMessage(data)
9.
10.     def connectionLost(self, reason):
11.         self.factory.clients.remove(self)
12.
13.     def connectionMade(self):
14.         self.factory.clients.add(self)
```



Y a continuación:

```
1. class ClientFactory(protocol.Factory):
2.
3.     def __init__(self):
4.         # Lista de los clientes conectados.
5.         self.clients = set()
6.
7.     def buildProtocol(self, addr):
8.         return ClientProtocol(self)
9.
10.    def sendMessage(self, message):
11.        """Enviar mensaje a todos los clientes."""
12.        for client in self.clients:
13.            client.transport.write(message)
```

Por último, iniciamos un servidor TCP en el puerto 9998, pero envolviendo nuestra clase `ClientFactory` dentro de `WebSocketFactory` para soportar el protocolo WebSocket.

```
1. endpoints.serverFromString(reactor, "tcp:9998").listen(
2.     WebSocketFactory(ClientFactory()))
3. reactor.run()
```

Código completo

Descarga: [websocketchat.zip](https://www.recursepython.com/wp-content/uploads/2016/07/websocketchat.zip) (<https://www.recursepython.com/wp-content/uploads/2016/07/websocketchat.zip>).

wsclient.html

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2.     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es"
4.     lang="es">
5.     <head>
6.         <title>Chat</title>
7.         <meta http-equiv="Content-Type" content="text/html;
8.             charset=utf-8" />
9.         <script src="chat.js"></script>
10.     </head>
11.     <body>
12.         <div id="login">
13.             <label>Nombre: </label> <input autocomplete="off"
14.                 type="text" id="username" value="Usuario" />
15.             <button onclick="loadChat();">Ingresar</button>
16.         </div>
17.         <div id="chat" hidden>
18.             <div id="messages" style="width: 600px; height:
19.                 300px; overflow-y: scroll; background-color: #D4FCFF;
20.                 border: 1px solid #CACACA; padding: 10px;">
21.                 Aún no te has conectado al chat.
22.             </div>
23.             <input type="text" style="width: 530px; height: 30px;
24.                 padding-left: 5px; margin: 0px;" id="message"
25.                 onkeyup="onKeyUp(event);" />
26.             <button style="width: 81px; height: 36px; padding:
27.                 0px; margin: 0px;" onclick="sendMessage();">Enviar</button>
28.             <br /><br />
29.             <button onclick="closeChat();">Salir</button>
```

```

23.     </div>
24.     <script>checkSupport();</script>
25. </body>
26. </html>

```

chat.js

```

1.  var ws;
2.  var username;
3.
4.  function onKeyUp(event)
5.  {
6.      if (event.keyCode == 13)
7.          sendMessage();
8.  }
9.
10. function sendMessage()
11. {
12.     message = document.getElementById("message");
13.     if (message.value)
14.     {
15.         ws.send("<strong>" + username + "</strong>: " +
message.value);
16.         message.value = "";
17.     }
18.     message.focus();
19. }
20.
21. function checkSupport()
22. {
23.     if (!("WebSocket" in window))
24.     {
25.         document.getElementById("login").innerHTML = "Este
navegador no soporta WebSockets.";
26.     }
27. }
28.
29. function loadChat()
30. {
31.     username = document.getElementById("username").value;
32.     if (!username)
33.         return;
34.
35.     document.getElementById("login").hidden = true;
36.     document.getElementById("chat").hidden = false;
37.
38.     messages = document.getElementById("messages");
39.     messages.innerHTML = "";
40.
41.     ws = new WebSocket("ws://localhost:9998");
42.
43.     ws.onopen = function()
44.     {
45.         ws.send(username + " ha ingresado al chat.");
46.     };
47.
48.     ws.onclose = function()
49.     {
50.         chat.innerHTML = "Se ha perdido la conexión."
51.     };
52.
53.     ws.onmessage = function(evt)
54.     {
55.         messages.innerHTML += evt.data + "<br>";

```



```

56.         messages.scrollTop = messages.scrollHeight;
57.     };
58. }
59.
60. function closeChat()
61. {
62.     ws.send(username + " se ha desconectado.");
63.     ws.close();
64. }

```

wsserver.py

```

1.  #!/usr/bin/env python
2.  # -*- coding: utf-8 -*-
3.
4.  from twisted.internet import protocol, reactor, endpoints
5.  from txws import WebSocketFactory
6.
7.
8.  class ClientProtocol(protocol.Protocol):
9.
10.     def __init__(self, factory):
11.         self.factory = factory
12.
13.     def dataReceived(self, data):
14.         self.factory.sendMessage(data)
15.
16.     def connectionLost(self, reason):
17.         self.factory.clients.remove(self)
18.
19.     def connectionMade(self):
20.         self.factory.clients.add(self)
21.
22.
23.  class ClientFactory(protocol.Factory):
24.
25.     def __init__(self):
26.         self.clients = set()
27.
28.     def buildProtocol(self, addr):
29.         return ClientProtocol(self)
30.
31.     def sendMessage(self, message):
32.         for client in self.clients:
33.             client.transport.write(message)
34.
35.
36.  endpoints.serverFromString(reactor, "tcp:9998").listen(
37.      WebSocketFactory(ClientFactory()))
38.  reactor.run()

```

Artículos relacionados

- [Tareas en segundo plano con PyQt/PySide](https://recursospython.com/guias-y-manuales/tareas-en-segundo-plano-con-pyqt/)
(<https://recursospython.com/guias-y-manuales/tareas-en-segundo-plano-con-pyqt/>)
- [Desarrollando una API REST con Twisted Klein](https://recursospython.com/guias-y-manuales/api-rest-twisted-klein/)
(<https://recursospython.com/guias-y-manuales/api-rest-twisted-klein/>)



- Twisted: web y base de datos
(<https://recursospython.com/guias-y-manuales/twisted-web-y-base-de-datos/>)
- Twisted – Arquitectura del framework de red más popular
(<https://recursospython.com/guias-y-manuales/twisted-arquitectura-del-framework-de-red-mas-popular/>)
- Actualizar contenido vía AJAX
(<https://recursospython.com/guias-y-manuales/actualizar-contenido-via-ajax/>)

Donar ❤️

¿Te gusta nuestro contenido? ¡Ayúdanos a seguir creciendo con una donación ([/donar](#))!

Entrada publicada en

Códigos de fuente (<https://recursospython.com/category/codigos-de-fuente/>)

Guías y Manuales (<https://recursospython.com/category/guias-y-manuales/>) con las etiquetas `javascript` (<https://recursospython.com/tag/javascript/>)

`twisted` (<https://recursospython.com/tag/twisted/>)

`txws` (<https://recursospython.com/tag/txws/>)

`websockets` (<https://recursospython.com/tag/websockets/>)

◀ Animación con PyGame + Exportarla como GIF

(<https://recursospython.com/guias-y-manuales/animacion-pygame-exportarla-gif/>)

Selector de idioma con web2py ▶

(<https://recursospython.com/codigos-de-fuente/selector-idioma-web2py/>)

8 comentarios.



Andres Niño says:

octubre 11, 2016 at 5:35 pm

(<https://recursospython.com/codigos-de-fuente/chat-via-web-websockets-twisted/#comment-261>)



Yo de nuevo: hay alguna forma de imprimir los nombres de los usuarios (clientes) desde el servidor (wsserver.py) para evitar que estos se repitan.

Gracias.

[Responder](#)



Recursos Python says:

[octubre 12, 2016 at 12:59 pm](#)

<https://recursospython.com/codigos-de-fuente/chat-via-web-websockets-twisted/#comment-262>

Hola. Sí, es posible, el problema con el ejemplo de este artículo es que el servidor no conoce más que los mensajes que envían los clientes (el usuario forma parte del mismo mensaje). Para evitar la repetición de nombres, el servidor debería tener una lista con cada uno de ellos. Para eso, el cliente tendría que enviar su nombre al servidor inmediatamente luego de establecer la conexión; el servidor chequearía la lista y tomaría la acción correspondiente.

Si tienes problemas para implementarlo te invito a que pases por el foro y lo veamos con mayor detalle.

Saludos.

[Responder](#)



Andres Niño says:

[octubre 8, 2016 at 1:02 am](#)

<https://recursospython.com/codigos-de-fuente/chat-via-web-websockets-twisted/#comment-254>



Excelente post, pero tengo una inquietud; que requisitos se deben cumplir para que el chat funcione cuando los usuarios establecen la conexión desde diferentes redes, o mejor conocida como conexión fuera de LAN.

Gracias

[Responder](#)



Recursos Python says:

[octubre 8, 2016 at 7:45 pm](#)

<https://recursospython.com/codigos-de-fuente/chat-via-web-websockets-twisted/#comment-255>

Hola Andres, me alegro que te haya servido. No se requiere nada adicional, simplemente que el puerto en el que escucha el servidor Twisted (9998 en el ejemplo) esté abierto para conexiones TCP. Una vez hecho esto, tendrás que reemplazar «localhost» por tu IP pública en la URI del WebSocket (`ws://localhost:9998`).

Un saludo.

[Responder](#)



Andres Niño says:

[octubre 9, 2016 at 12:35 am](#)

<https://recursospython.com/codigos-de-fuente/chat-via-web-websockets-twisted/#comment-256>

Muchas gracias por su pronta respuesta, me ha funcionado.

[Responder](#)



Recursos Python

says:

[octubre 9, 2016 at 7:33 pm](#)
<https://recursospython.com/codigos-de-fuente/chat-via-web-websockets-twisted/#comment-257>

¡De nada!

[Responder](#)



David says:

[marzo 31, 2020 at 5:12 pm](#)
<https://recursospython.com/codigos-de-fuente/chat-via-web-websockets-twisted/#comment-663>

Si accedo al cliente mediante el archivo html me va todo perfecto, pero instando acceder con la ruta ws://localhost:9998 me pone que el servidor no está activo (cuando sí lo está). A que se puede deber esto?.
Gracias.

[Responder](#)



Recursos Python

says:

[abril 2, 2020 at 12:49 pm](#)
<https://recursospython.com/codigos-de-fuente/chat-via-web-websockets-twisted/#comment-666>



Hola David, `ws://localhost:9998/` es la dirección de un WebSocket, no deberías poder acceder directamente a través del navegador.

[Responder](#)

Deja una respuesta

Comentario *

Nombre *

Email *

Publicar el comentario

© 2013 - 2023

¡Suscríbete a nuestra newsletter! (<https://www.recursospython.com/newsletter/>)

[Políticas de Uso y Privacidad \(https://www.recursospython.com/politicas-de-uso-y-privacidad/\)](https://www.recursospython.com/politicas-de-uso-y-privacidad/)

En inglés: [Python Assets \(https://pythonassets.com/\)](https://pythonassets.com/)



(<https://creativecommons.org/licenses/by-nc/3.0/deed.es>)

