Revista Cubana de Ciencias Informáticas
Vol. 7, No. 3, Julio - Septiembre, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. 37-44
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Técnicas de programación
Recibido: 6/10/2012 | Aceptado: 6/09/2013

El *framework* jWebSocket y su interfaz de aplicaciones para el trabajo con tarjetas inteligentes

The jWebSocket framework and application interface for working with smart cards

Ander Sánchez Jardines[1*], Alexander López Pupo[1], Martha Rodríguez Freire[2]

[1] Departamento de Tarjetas Inteligentes. Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½ Torrens, Boyeros, La Habana, Cuba. C.P.: 19370.
[2] CEIGE. Departamento de Informatización de entidades. Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km 2 ½ Torrens, Boyeros, La Habana, Cuba. C.P.: 19370.

[*]Autor para correspondencia: ajardines@uci.cu

Resumen

El jWebSocket es un marco de trabajo y a la vez un servidor de aplicaciones para la plataforma Java orientado al desarrollo de soluciones basadas en WebSockets, que gocen de altos niveles de velocidad, escalabilidad y seguridad. Sus grandes potencialidades en cuanto al soporte concurrente y su licencia de *software* libre hacen que sea adoptado por una gran comunidad de desarrolladores. El API[1] de tarjetas inteligentes es una extensión para el marco de trabajo jWebSocket, que le permite a este último soportar los requerimientos necesarios para desarrollar *software* empresarial y realizar disímiles operaciones con las tarjetas inteligentes, obteniendo resultados favorables para el uso de cualquier navegador y brindando flexibilidad y tiempo real, características que distinguen a la web. El presente artículo expone un conjunto de ventajas y características del marco de trabajo jWebSocket y de su API para el intercambio con las tarjetas inteligentes, explicando conceptos relacionados con el tema, revelando las soluciones más destacadas en el mundo.

Palabras clave: API de tarjetas inteligentes, jWebSocket, servidor de aplicaciones, tarjetas inteligentes, WebSocket.

Abstract

The jWebSocket is a framework and an application server for the Java platform aimed at developing solutions based on WebSockets, which enjoy high levels of speed, scalability and security. Its great potential in terms of concurrent support and free software license makes it adopted by a large community of developers. The API[1] smart card is an extension for jWebSocket framework, which allows the latter support the requirements needed to develop enterprise software and dissimilar conduct operations with smart cards, with favorable results for the Using any browser, providing flexibility and real-time features that distinguish the web. This article presents a set of advantages and

[1]**Interfaz de programación de aplicaciones** (**IPA**) o API (*ApplicationProgramming Interface*) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

features of the framework and its API jWebSocket for exchange with smart cards, explaining concepts related to the topic, revealing the most prominent solutions in the world.
Keywords:*Application server,smart card API, jWebSocket, smart cards, WebSocket.*

Introducción

El surgimiento de Internet en la segunda mitad del pasado siglo ha revolucionado la manera de vivir y pensar del ser humano. En la actualidad se brindan un sinfín de servicios en línea por parte de instituciones, gobiernos o grandes empresas; a los cuales el usuario o beneficiario accede desde su computador. Entre estos se pueden destacar el gobierno en línea, las pasarelas de pago, los servicios de correo electrónico, la reservación de servicios hoteleros o de viaje, la certificación de documentos, la emisión de declaraciones de impuestos o la tramitación de documentos oficiales de una nación determinada.

Actualmente se registra una tendencia al uso de dispositivos inteligentes que facilitan y aseguran el proceso de identificación. Se han desarrollado tecnologías como las tarjetas inteligentes para garantizar una verificación más exacta y confiable de la identidad del usuario que solicita determinado servicio (Betarte, 2001).

Del mismo tamaño que una tarjeta de crédito, una tarjeta inteligente contiene un circuito integrado en su cuerpo de plástico que la convierte en un ordenador portable. Al contrario que las tarjetas de banda magnética, las tarjetas inteligentes tienen capacidad de procesar datos y proporcionar protección física (hardware) de los datos que almacenan. Las tarjetas inteligentes pueden transferir datos a través de contactos en su superficie o en las llamadas tarjetas inteligentes sin contactos, mediante campos electromagnéticos (Pineda, 2003; Álvarez, 2010).

Al aumentar exponencialmente los usuarios en línea y el acceso a la web desde distintos dispositivos móviles, surgen nuevas tendencias en la comunicación web tales como: mayor interactividad, operabilidad, movilidad y tiempo real.

Para brindar solución a la comunicación web en tiempo real y a las tendencias mencionadas anteriormente surge el protocolo WebSockets, tecnología que proporciona un canal de comunicación bidireccional y *full-duplex* sobre un único enchufe (socket) TCP (Startseite, 2009). Esta tecnología está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor.

Según (Oscar Cassetti, 2010), los principales servidores que soportan WebSockets para el desarrollo de aplicaciones hoy día son, la pasarela WebSockets deKaazing (Kaazing, 2008), JettyWebSocketServlet (Eclipse, 2009), Socket.IO (socket.io, 2008), django-websocket del proyectoPython (python, 2007) y jWebSocket (jwebsocket.org, 2008).

Este artículo persigue como objetivo introducir algunas ventajas y características asociadas al frameworkjWebSocket que facilitan el uso del mismo para la ejecución de aplicaciones de tarjetas inteligentes.

Materiales y métodos

Conceptos asociados al tema abordado

La unidad de comunicación entre un lector y una tarjeta es llamada unidad de datos de protocolo de aplicación (*ApplicationProtocol Data Unit)*, su estructura está definida en el estándar ISO 7816, existiendo dos tipos de categorías de APDU, APDU *Command* (Comando APDU) y APDU *Response* (APDU Respuesta) (ISO Organization, 2005).

El objetivo, a manera general, de las 14 partes que componen el estándar 7816 es lograr la interoperabilidad entre distintos fabricantes de tarjetas inteligentes y lectores de las mismas, en lo que respecta a características físicas, comunicación de datos y seguridad. Estos estándares son basados en los ISO 7810 e ISO 7811, los cuales definen características físicas de tarjetas de identificación (Estándar ISO/IEC 7816, 2006).

En la actualidad son varios los sistemas operativos que existen para las tarjetas inteligentes. Entre los más utilizados se encuentra *JavaCard*, definido como una tecnología que permite a las tarjetas inteligentes y otros dispositivos con memoria muy limitada ejecutar pequeñas aplicaciones, llamadas *applet*s (Marquez, 2006).
Los *applet* son aplicaciones que corren en las tarjetas inteligentes y comienzan su ciclo de vida al ser correctamente cargados en la memoria de las mismas (Effing, 2003).
Es importante mencionar que los *applet* hechos en *javacard* están soportados sobre la plataforma Java, creada por la empresa Sun Microsystems en 1995. Es la tecnología subyacente que permite el uso de programas punteros, como herramientas, juegos y aplicaciones de negocios. Java se ejecuta en más de 850 millones de ordenadores personales de todo el mundo y en miles de millones de dispositivos, como dispositivos móviles y aparatos de televisión.(Gómez, 2010)
El ***middleware*** es un *software* destinado a proporcionar conectividad, interoperalidad o integración entre diferentes aplicaciones, normalmente distribuidas, y en el peor de los casos, sobre recursos heterogéneos. Funciona como una capa de abstracción de *software* distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red). El middleware abstrae de la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando una API para la fácil programación y manejo de aplicaciones distribuidas (Karl, 2011).
El protocolo Websocket permite realizar conexiones bidireccionales entre un cliente y un servidor. El objetivo de esta tecnología es proveer un mecanismo para aplicaciones basadas en navegadores que necesitan comunicación bidireccional con el servidor en vez de tener que realizar múltiples conexiones HTTP (Hybi, 2011; Furukawa, 2011).
Las conexiones entre un navegador web y un servidor basadas en el protocolo WebSocket, producen un consumo de ancho de banda de 1/50 comparado con el consumo del protocolo HTTP y reduce la latencia en el orden de 1/3 (Hybi, 2011). Websocket es la tecnología estándar para la web que permite que se puedan establecer conexiones en tiempo real entre un cliente y un servidor.

JWebSocketfue fundado por Alexander Schulze a finales de 2009 y respaldado por la empresa InnotradeGmbH con sede en Alemania. Es un marco de trabajo y a la vez un servidor de aplicaciones para la plataforma Java orientado al desarrollo de soluciones basadas en Websockets, las aplicaciones derivadas gozan de altos niveles de velocidad, escalabilidad y seguridad. Se pueden implementar aplicaciones HTML5, ofrece amplia gama de funcionalidades desde el intercambio de bajo nivel basado en *tokens*, hasta la sincronización de interfaz gráfica de usuario, llamadas a procedimiento remoto (Schulze, *et al.*, 2011). Sus grandes potencialidades en cuanto al soporte concurrente y su licencia LGPL2 hacen que sea adoptado por una gran comunidad de desarrolladores.
JWebSocket establece un modelo de *token*. Los *tokens* son datos abstractos que a través de una estructura jerárquica y una API proporcionan métodos de acceso a los contenidos. Con el objetivo de realizar una abstracción en la manipulación de los diferentes formatos, el marco de trabajo convierte los paquetes de datos entrantes y salientes en

[2] http://www.gnu.org/licenses/lgpl.html

*token*s. El cliente nativo soporta el intercambio de paquetes en los formatos JSON, XML y CSV, que en entornos específicos se pueden utilizar sin la necesidad de manejarlos a través de *token*s (Schulze, 2008).

El cliente jWebSocket tiene una arquitectura de plug-in que permite aumentar con facilidad sus funcionalidades. En caso de que los servidores soporten de manera nativa Websocket, como el caso de Jetty o GlassFish, se incluyen las funciones de comunicación del marco de trabajo jWebSocket, pero los motores internos se apagan y el anfitrión se utiliza. Esto asegura que no haya mecanismos de seguridad adicionales (Schulze, *et al.*, 2011).

Otras soluciones

CoesyseGov 2.0 producido por Gemalto[3], permite un servicio de identificación electrónica mediante tarjetas inteligentes basado en la *web*, en vez de un *software* basado en un cliente de autenticación de instalación local. Esta solución evita la administración de un *software middleware* en el cliente, toda la funcionalidad requerida se centraliza en un servidor. CoesyseGov 2.0 se presenta para solucionar el problema de emisión de certificados, pues la generación de llaves y solicitud/carga de certificados necesitan llevarse a cabo en las tarjetas inteligentes en modo de post-emisión. No requiere *software* en la computadora del cliente, simplificando el despliegue de servicios y potenciando una mayor asimilación. Entre sus características principales están: servicios de conectividad de tarjetas inteligentes, servicio de autenticación, federación de identidad (Gemalto, 2010).

SConnect es una extensión para los navegadores más importantes, es compatible con los sistemas operativos Windows, Mac OSX y Linux. Su objetivo principal es el de proporcionar un puente de conexión entre el Java Script, que corre en la página web de un navegador y la tarjeta inteligente, permitiendo la conectividad entre estas últimas aplicaciones y los servicios *web* (Gemalto Security, 2008).

SConnect consiste en dos partes: una extensión del navegador web que conecta con la capa PC/SC estándar del ordenador, conectando una página web con una tarjeta inteligente, que se comunica con un ordenador *host* vía PC/SC y una librería Java Script que permite a los desarrolladores de aplicaciones web tener acceso a tarjetas inteligentes mediante SConnect. El aspecto de conectividad de SConnect es la clave en la innovación de la solución de CoesyseGov 2.0.

Resultados y discusión

Características del API de Tarjetas Inteligentes perteneciente al framework jWebSocket

La manera tradicional de interactuar con las tarjetas inteligentes es a través de capas o librerías de *software*, técnicamente conocidas como *middleware*, que normalmente corren en el cliente y hacen función de intermediarios entre diversas aplicaciones y los lectores de tarjetas. El manejo de las tarjetas mediante el uso de *middleware*s que se ejecuten en el cliente trae consigo riesgos considerables. Algunas de las desventajas que trae es el caso de las actualizaciones del *software* o la incorporación de nuevas funcionalidades a estas librerías, habría que distribuirlas por todos los clientes de un sistema dado o publicarlos en un sitio *web* para que sean descargados a través de la red. Esto, además de ser incómodo para el usuario, implica que deben tener ciertos conocimientos para efectuar las actualizaciones y además poseer una serie de permisos en el manejo de los recursos de la computadora.

La API para la gestión de tarjetas inteligentes en aplicaciones *web* desarrolladas con el marco de trabajo jWebSocket, permitirá obtener los lectores disponibles conectados a la estación cliente, establecer la comunicación con los lectores de tarjetas, notificar el estado de la conexión con la tarjeta, controlar el intercambio de comandos y respuestas APDU entre el *middleware* por el lado del servidor y la tarjeta inteligente en el cliente, así como ejecutar una función

[3] Líder mundial en la venta de tarjetas inteligentes y las aplicaciones asociadas

correspondiente a un determinado *middleware* en el servidor, permitiendo comenzar de esta manera la secuencia de los comandos APDU.

Esta API brindará la posibilidad de administrar desde el servidor los *middleware*s, lo que traerá consigo diferentes beneficios como son: la solución de procesos de instalación, flexibilidad en la actualización de las funciones del *middleware* e interoperabilidad con varias tarjetas inteligentes, centralización en el servidor de todas las funcionalidades requeridas, simplificando el despliegue de servicios y potenciando una mayor asimilación. Permitiendo además que la comunicación no se establezca siempre ante una solicitud del usuario, sino que sea proactiva, que el servidor pueda comunicarse con ellos sin una acción precedente. Esto se logrará haciendo uso del marco de trabajo jWebSocket, una nueva tecnología orientada al desarrollo de aplicaciones *web* basadas en Websocket. Actualmente existen soluciones que manejan el *middleware* en el lado del servidor pero ninguna usa como protocolo de comunicación Websocket.
El nivel de escalabilidad de las conexiones soportadas por los servidores jWebSocket permite alta concurrencia de usuarios. Este elemento no solo garantiza realizar procesos en tiempo real, sino poder garantizar un alto número de usuarios utilizando un mismo servicio en el mismo instante de tiempo.

Arquitectura del API de Tarjetas Inteligentes

La arquitectura que propone el marco de trabajo jWebSocket para el trabajo con tarjetas inteligentes es la arquitectura en capas, dicha arquitectura tiene como objetivo primordial separar la lógica de negocios de la lógica de diseño. La principal ventaja de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que ocurra algún cambio sólo afectará dicho nivel, logrando obviar las demás capas del sistema. Permite además distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles. Esta arquitectura implementa de manera sencilla varios *middleware* del lado del servidor, esto favorece a los usuarios de tarjetas ya que no tienen necesidad de descargar ningún tipo de herramienta para de una forma asequible y segura acceder a información relevante de la tarjeta desde su propio computador utilizando la *web*.

Componentes relacionados con el API

En el diagrama de componentes que se muestra a continuación (ver Figura), se describen los elementos físicos y relaciones del marco de trabajo jwebsocket, resaltando con un color más oscuro, el API de extensión del administrador de *middleware* que soporta la lógica para el trabajo con tarjetas inteligentes.

Revista Cubana de Ciencias Informáticas
Vol. 7, No. 3, Julio - Septiembre, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. 37-44
http://rcci.uci.cu

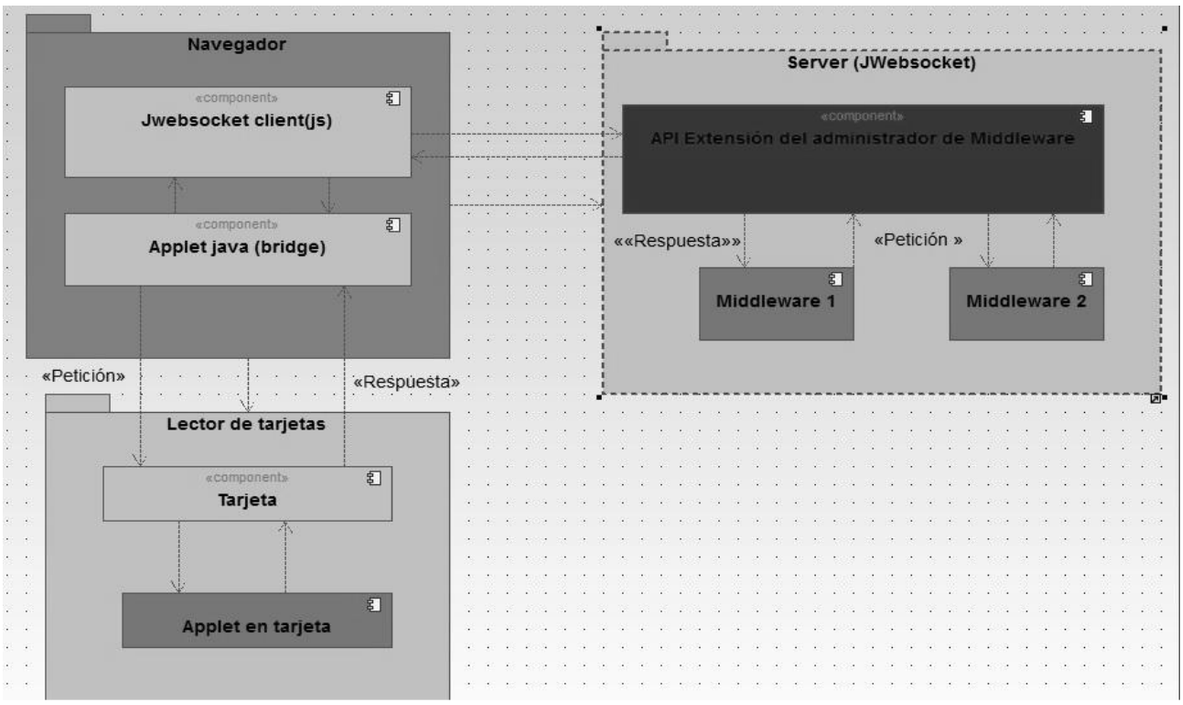

Figura. Diagrama de componentes.

Descripción:

- El cliente debe contar con un navegador que soporte el protocolo Websocket.
- Esta arquitectura permite tanto al cliente como al servidor iniciar la comunicación.
- El cliente de jWebSocket se comunica con el servidor de jWebSocket, enviando una solicitud al componente *Middleware* Operation Manager que gestiona dicha solicitud, enviando una respuesta a la tarjeta inteligente. Esta respuesta pasa por el cliente de jWebSocket y el *applet* de Java quien funciona como puente entre el cliente y el lector de tarjeta, la tarjeta inteligente procesa la respuesta y envía otra al *middleware* en el servidor.
- En el lector se inserta la tarjeta inteligente, en ella se encuentran todas las aplicaciones.

La API permite desarrollar aplicaciones *web* en tiempo real haciendo uso de los beneficios que aportan las tarjetas inteligentes, ya que estas ofrecen una mayor seguridad en la comunicación, transferencia y almacenamiento de la información en la *web*. Dicha API emplea los recursos propios del modelo de eventos (EventModel), recursos que facilitan la comunicación con los eventos desencadenados por el cliente y extiende y particulariza algunas de las características del mismo.

EventModel es un objeto que cumple con el patrón observable y representa el núcleo EventsPlugIn. Contiene los oyentes de aplicación, filtros, extensiones, fábrica de eventos, el controlador de notificación de eventos S2C y los manejadores de excepciones. Una aplicación EventsPlugIn es de hecho una instancia de objeto EventModel.
El EventModel tiene dentro de sus funcionalidades fundamentales, procesar evento, que es encargado del manejo de los eventos que arriban al servidor y los métodos de lectura y escritura de *tokens*y mensajes S2C[4] y C2S[5].

Utilizando las funcionalidades del EventModella API permite obtener los lectores disponibles conectados a la estación cliente, establecer la comunicación con los lectores de tarjetas, notificar el estado de la conexión con la tarjeta, controlar el intercambio de comandos y respuestas APDU entre el *middleware* por el lado del servidor y la

[4]Server to client
[5]Client to server

tarjeta inteligente en el cliente, así como ejecutar una función correspondiente a un determinado *middleware* en el servidor, permitiendo de esta manera comenzar la secuencia de los comandos APDU.

Esta API dota al marco de trabajo jWebSocket de un conjunto de librerías que permite desarrollar aplicaciones web haciendo uso de las tarjetas inteligentes para los distintos sectores de la sociedad, entre las que se puede mencionar el Comercio Electrónico en el uso de los monederos electrónicos e inversiones bancarias, así como en el Control de Acceso e Identificación Física. Otro de sus importantes usos en la actualidad, es su vinculación a la firma digital de documentos, o en el área de la Salud para la identificación de pacientes y control de los datos del historial clínico. De igual forma se hace uso en el sector público para realizar votos electorales, para el Transporte, en el pago de la cuota de autobús sin necesidad de usar efectivo o monedas, o como un accesorio más de las personas, en licencia de conducción o documento de identificación mediante los certificados contenidos en la memoria no volátil del chip. Por el auge que han tenido y los disimiles beneficios que ofrece, surge como tendencia el uso de dispositivos inteligentes, como las tarjetas para facilitar y asegurar el proceso de identificación en la web, garantizando una verificación más exacta y confiable de la identidad del usuario que solicita determinado servicio en la *web*.

Conclusiones

- Esta API, a pesar de tener un conjunto de funcionalidades incipientes, proporciona la base para nuevas implementaciones de librerías y extensiones, permitiendo la gestión de tarjetas en aplicaciones webdesarrolladas con el marco de trabajo jWebSockety a su vez garantizando un aumento de los niveles de seguridad y usabilidad.
- Dos de las características más destacadas enla API son: la flexibilidad y potencialidad de las librerías desarrolladas para la gestión de tarjetas, garantizando que su código fuente y documentación poseen la calidad requerida para su posterior continuidad y entendimiento por desarrolladores menos adiestrados en los temas de Jwebsocket y tarjetas inteligentes.
- Sobre las soluciones que se describen de manera sintéticaen el artículo, se puede concluir que, además de ser productos propietarios con muy alto costo, tienen poca claridad en el mecanismo de incorporación de nuevos *middleware*s.

Referencias

- ALEXANDER SCHULZE, ROLANDO SANTAMARÍA MASÓ.*WebSocket für alle.* Alemania: s.n., 2011.
- BETARTE, GUSTAVO. 2001.*Programación de JavaCards.* España: s.n., 2001.
- ECLIPSE.[en línea] 2009 [Consultado el: 27 de abrilde 2013] Disponible en: [http://www.eclipse.org/jetty/].
- EFFING, WOLFGANG RANKL and WOLFGANG. 2003.*Smart Card HandBook.* New York: John Wiley & Sons Ltd, Baffins Lane, Chichester, 2003.
- ESTÁNDAR ISO/IEC 7816. 2006.*Estándares ISO/IEC 7816.* 2006.
- FURUKAWA, Y. 2011.Web-Based Control Application Using Websocket. *Web-Based Control APPLICATION USING WEBSOCKET*. [en línea] 2011 [Consultado el: 10 de marzo de 2013] Disponible en: [http://accelconf.web.cern.ch/AccelConf/icalepcs2011/papers/wemau010.pdf].
- GEMALTO. 2010. Coesys eGov 2.0 V3 . *Coesys eGov 2.0 V3* . [en línea] 2010 [Consultado el: 10 de febrero de 2013].Disponible en: [http://www.gemalto.com/govt/coesys/coesys_egov2_0_version3.html].
- GEMALTO SECURITY. Sconect. *Sconect.* [en línea] 2008 [Consultado el: 20 de abril de 2013].Disponible en: [http://www.gemalto.com/investors/agm/agm-2008/download/annual_report_2007.pdf].

- GÓMEZ, CÉSAR.[en línea] 2010 [Consultado el: 20 de diciembre de 2011].Disponible en: [http://www.osupiita.com/index.php/proyectos/micro-edicion].
- HYBI. The WebSocket protocol. *The WebSocket protocol.* [en línea] 2011 [Consultado el: 1 de diciembre de 2011] Disponible en: [http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-17].
- ISO Organization. 2005.*ISO/IEC 7816-4.* 2005.
- jwebsocket.org. 2008. jwebsocket. *jwebsocket.* [en línea] 2008 [Consultado el: 25 de abril de 2013].Disponible en: [http://jwebsocket.org/].
- KAAZING. 2008. Kaazing WebSocket Gateway. *Kaazing WebSocket Gateway.* [en línea] 2008 [Consultado el: 21 de abril de 2013].Disponible en: [http://kaazing.com/products/kaazing-websocket-gateway].
- KARL. Middleware. *Middleware.* [en línea] 2011 [Consultado el: 21 de agosto de 2012].Disponible en: [http://www.buenastareas.com/ensayos/Middleware/2031177.html].
- MARQUEZ TORRES, JOAQUIN.Disponible en: [http://e-archivo.uc3m.es. *http://e-archivo.uc3m.es*]. [en línea] 2006. [Consultado el: 17 de marzo de 2012].Disponible en: [http://e-archivo.uc3m.es/bitstream/10016/781/1/Tesis_Doctoral-Joaquin_Torres_Marquez.pdf].
- OSCAR CASSETTI, SATURNINO LUZ. 2010.*The WebSocket API as Supporting Technology For Distributed and Agent-Driven Data Mining.* Irlanda: s.n., 2010.
- PINEDA, ESTEBAN, LUCÍA.*Emulador de SAT.* España: s.n., 2003.
- PYTHON.[en línea] 2007 [Consultado el: 22 de abril de 2013].Disponible en: [https://pypi.python.org/pypi/django-websocket].
- SANTIAGO, IGNACIO ÁLVAREZ. 2010.*Gestión de SmartCards mediante PKCS#11.* España : s.n., 2010.
- SCHULZE, ALEXANDER. 2008.*Framework Approach for WebSockets.* Alemania: s.n., 2008.
- socket.io. 2008.[en línea] 2008 [Consultado el: 12 de febrero de 2013].Disponible en: [http://socket.io/].
- ZUR STARTSEITE. Formatos y estándares de software . *Formatos y estándares de software.*[en línea] 2009 [Consultado el: 21 de enero de 2012]. Disponible en: [http://www.es.hukol.net/themenreihe.p?c=Formatos_y_est%C3%A1ndares_de_software].