

# Universidad de las Ciencias Informáticas

## Facultad Regional Mártires de Artemisa



### Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

#### **Título:**

**Librería cliente C Sharp para desarrollar aplicaciones con  
el marco de trabajo jWebSocket**

#### **Autor:**

Rolando Betancourt Toucet

#### **Tutor:**

Msc. Yamila Vigil Regalado


#### **Cotutor:**

Lic. Gilberto Ramón Justiniani Fernández

#### **Consultante:**

Dr. C.T. Ingeniero Raúl Rene Crespo Heredia

Artemisa, Cuba, Junio 2012



*El hombre  
inteligente no es el  
que tiene muchas  
ideas, sino el que  
sabe sacar provecho  
de las pocas que  
tiene.*

*Atónimo*

# DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Desarrollo de la Facultad Regional “Mártires de Artemisa” de la Universidad de las Ciencias Informáticas; así como a dicha universidad para que hagan el uso que estimen pertinente con el mismo. Para que así conste firmo la presente declaración a los \_\_\_\_\_ días del mes de \_\_\_\_\_ del año 2012.

---

**Rolando Betancourt Toucet**

Firma del Autor

---

**Yamila Vigil Regalado**

Firma del Tutor

---

**Gilberto R Justiniani Fernández**

Firma del Cotutor

# AGRADECIMIENTOS

Agradezco a mi familia, por el apoyo incondicional que me han brindado durante todos estos años, en especial a mis padres y mi tía Arelys que siempre han estado presentes cuando los he necesitado.

Agradezco a todos mis amigos que me han acompañado a lo largo de estos cinco años de la carrera.

Agradezco a todos los profesores que de una manera u otra me han ayudado a realizar este sueño.

Por ultimo pero no menos importante un especial agradecimiento al equipo de desarrollo del proyecto jWebSocket de la Facultad Regional Mártires de artemisa de la Universidad de las Ciencias Informáticas.

# DEDICATORIA

Dedico este trabajo de diploma a mi familia que siempre me ha apoyado para que yo logre mis objetivos en la vida.

También le dedico este trabajo a mi novia que de una forma u otra siempre me ha brindado su apoyo.

Por último me dedico este trabajo, que significa para mí la culminación satisfactoria de cinco años de estudios, los mejores cinco años de mi vida.

# RESUMEN

jWebSocket es un marco de trabajo que utiliza el protocolo WebSocket para desarrollar aplicaciones en tiempo real para la Web. Este permite implementar las aplicaciones del lado del servidor empleando el lenguaje de programación Java<sup>1</sup>. Mediante este protocolo es posible brindar una arquitectura muy flexible, a través de la cual las aplicaciones del lado del cliente pueden ser desarrolladas en cualquier lenguaje de programación. En la actualidad este marco de trabajo sólo ofrece soporte para aquellas aplicaciones cliente que estén desarrolladas con los lenguajes Java o JavaScript<sup>2</sup>.

Hoy día los desarrolladores del lenguaje de programación C Sharp no pueden aprovechar las potencialidades de jWebSocket porque este no brinda soporte para una librería cliente en este lenguaje. Esto obliga a los desarrolladores a invertir tiempo y recursos en implementar el protocolo WebSocket para lograr la conexión y transferencia de datos a bajo nivel con el servidor, alejándose de la lógica de la aplicación. Este hecho provoca una disminución de los niveles de productividad y confiabilidad de las aplicaciones desarrolladas.

Al concluir esta investigación se obtuvo una librería que cumple con todas especificaciones del cliente para la comunicación entre una aplicación cliente y el marco de trabajo jWebSocket. Teniendo en cuenta los resultados satisfactorios de los casos de pruebas funcionales realizados, obtenidos el aval de certificación de calidad y el aval del experto Alexander Schulze la librería cliente C Sharp quedó disponible para la versión 1.0.

---

<sup>1</sup> [http://www.cad.com.mx/historia\\_del\\_lenguaje\\_java.htm](http://www.cad.com.mx/historia_del_lenguaje_java.htm)

<sup>2</sup> <http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/>

# ÍNDICE GENERAL

<b>INTRODUCCIÓN.....</b>	<b>1</b>
<b>Capítulo 1. Fundamentación Teórica.....</b>	<b>9</b>
1.1    Conceptos Asociados al Dominio del Problema.....	9
1.2    Análisis de Soluciones Existentes .....	11
1.3    Metodología usada para el desarrollo de la solución .....	13
1.4    Lenguaje de Modelado de Software.....	16
1.5    Herramienta CASE .....	17
1.6    Herramientas y Tecnologías Utilizadas .....	19
1.6.1    Entorno integrado de desarrollo – IDE.....	19
1.6.2    Marco de Trabajo.....	20
1.6.3    Herramientas de Control de Versiones .....	21
1.6.4    Herramientas de Control de Versiones del lado del cliente .....	24
<b>2.    Capítulo 2. Características, Análisis y Diseño del Sistema.....</b>	<b>27</b>
2.1    Propuesta de Solución .....	27
2.2    Planificación del Proyecto por Roles .....	28
2.3    Modelo de Historias de Usuario del Negocio.....	30
2.4    Lista de Reserva del Producto (LRP) .....	31
2.5    Historias de Usuario y Tareas de Ingeniería .....	33
2.6    Plan de Releases .....	42
2.7    Descripción de la Arquitectura.....	43
2.8    Diseño con Metáforas.....	44
2.9    Diagrama de Componente .....	45
<b>3    Capítulo 3. Implementación y Validación del Sistema.....</b>	<b>49</b>
3.1    Diagrama de Despliegue .....	49
3.2    Validación de la Investigación .....	50
3.3    Aplicación demostrativa.....	51
3.4    Casos de Prueba.....	52

# ÍNDICE GENERAL

3.5	Certificación de calidad. ....	62
3.6	Valoración del Cliente.....	63
3.7	Resultados Obtenidos. ....	63
3.8	Funcionalidades Obtenidas. ....	64
3.9	Aporte Social y Económico.....	65
	<b>CONCLUSIONES.....</b>	<b>67</b>
	<b>RECOMENDACIONES .....</b>	<b>68</b>
	<b>BIBLIOGRAFÍA REFERENCIADA .....</b>	<b>69</b>



# ÍNDICE DE TABLAS Y FIGURAS

Fig. 1: Modelo de Historias de Usuario del Negocio.....	31
Fig. 2: Diagrama de Paquete de la librería cliente C Sharp.....	45
Fig. 3: Diagrama de Componentes de la librería cliente C Sharp.....	46
Fig. 4 Diagrama de Despliegue .....	49
Fig. 5 Aplicación demostrativa.....	52
Tabla 1: Planificación del proyecto por roles .....	28
Tabla 2: Lista de Reserva del Producto.....	32
Tabla 3: Historia de Usuario Establecer conexión .....	33
Tabla 4: Tarea de Ingeniería Crear el Handshake.....	34
Tabla 5: tarea de Ingeniería Procesar el Handshake .....	35
Tabla 6: Tarea de Ingeniería Establecer la Conexión.....	35
Tabla 7: Historia de Usuario Enviar Token .....	36
Tabla 8: Tarea de Ingeniería Enviar Token Binario .....	37
Tabla 9: Tarea de Ingeniería Enviar Token de Texto .....	37
Tabla 10: Tarea de Ingeniería Enviar Token Fragmentado .....	38
Tabla 11: Tarea de Ingeniería Enviar Paquete .....	38
Tabla 12: Historia de Usuario Recibir Token .....	39
Tabla 13: Tarea de Ingeniería Recibir Token .....	40
Tabla 14: Tarea de Ingeniería Recibir e Interpretar Fragmentos de Token .....	40
Tabla 15: Historia de Usuario Finalizar Conexión.....	41
Tabla 16: Tarea de Ingeniería Finalizar Conexión.....	41
Tabla 17: Plan de Releases.....	42
Tabla 18 Caso de Prueba JWS-01-01 .....	53
Tabla 19 Caso de Prueba JWS-01-02.....	54
Tabla 20 Caso de Prueba JWS-02-01 .....	56
Tabla 21 Caso de Prueba JWS-02-02.....	58
Tabla 22 Caso de Prueba JWS-03-01 .....	59
Tabla 23 Caso de Prueba JWS-04-01 .....	61

Desde su creación la Web se ha convertido en uno de los métodos que ofrece Internet para consultar e intercambiar información con mayor eficacia. Con el paso de los años lo que comenzó como un simple sistema de distribución de información se ha transformado en un medio de entretenimiento y comunicación social. Hoy en día la Web esta abarrotada de juegos en línea, redes sociales y aplicaciones para transferencia e intercambio de archivos. Es además un espacio para brindar servicios en el ámbito profesional tales como flujos de colaboración en línea, notificaciones de noticias y estados financieros entre otros.

Para lograr una mayor interactividad en la Web es preciso realizar notificaciones en tiempo real. Esto viene dado por la necesidad que tienen los usuarios de ser notificados sobre una acción determinada en el instante en que ocurre sin tener que realizar una solicitud previa al servidor. Es necesario que estos usuarios conozcan los cambios ocurridos en las aplicaciones que están suscritos tales como la llegada de nuevo correo, un cambio en la bolsa de valores o simplemente estar al tanto del estado del tiempo para una región determinada.

Como sucede con otras tendencias de innovación reciente, existen muchas definiciones para el modelo del tiempo real en la Web. Una de estas definiciones es la ofrecida por el líder del proyecto internacional WebSocket que plantea lo siguiente.” Tiempo real en la Web significa lograr dos elementos: el cliente recibe mensajes del servidor sin solicitud previa y los usuarios finales reciben actualizaciones de forma simultánea”. (Framework Approach for WebSockets, 2011)

*Hypertext Transfer Protocol* (HTTP<sup>3</sup>) fue intencionalmente diseñado como un simple protocolo de solicitud-respuesta. Este esquema no proporciona una comunicación bidireccional entre el cliente y el servidor. Esto ocasiona que si se genera un nuevo

---

<sup>3</sup> (Protocolo de transferencia de hipertexto), es el método más común de intercambio de información en internet, es el método mediante el cual se transfieren las páginas web a un ordenador.

evento en el servidor, este no es capaz de enviarle directamente un mensaje al cliente de manera asíncrona. En todos los casos el cliente debe realizar una solicitud, para así el servidor pueda enviar una nueva información, ocasionando demoras en la entrega de datos actualizados al usuario final. (Comet-HTML5-WebSocket, 2011)

Con el objetivo de resolver las limitaciones que presenta el protocolo HTTP se han creado diversas técnicas, tales como *polling*<sup>4</sup> y *long polling*<sup>5</sup>. Ambas se basan en realizar sondeos al servidor en intervalos de tiempo regulares con el objetivo de mantener actualizados los datos del lado del cliente. También puede variar la forma en que responde el servidor para optimizar el número de peticiones innecesarias. Estas técnicas continúan presentando limitaciones en cuanto al ancho de banda utilizado y la latencia que generan en la red.

Finalmente surge HTML5-WebSocket, un nuevo protocolo para dar solución a la comunicación web en tiempo real. Dicho protocolo cuenta con la aprobación de la *World Wide Web Consortium (W3C)*, el grupo *Web Hypertext Application Technology Working Group (WHATWG)* y el grupo *Hypertext Bidirectional (Hybi)* de la *Internet Engineering Task Force (IETF)*. Su primer borrador se publicó el 9 de enero del 2009 y ha ido evolucionando hasta la actualidad.

WebSocket<sup>6</sup> es un protocolo que fue diseñado para ser implementado en navegadores y servidores web, pero brinda la posibilidad de utilizarse por cualquier aplicación que cumpla con la arquitectura cliente – servidor. Para establecerse una conexión mediante WebSocket se debe realizar una negociación que es iniciada por el cliente, la cual debe ser respondida por el servidor de forma satisfactoria. Esta negociación consiste en una actualización del protocolo HTTP proporcionando un canal de comunicación bidireccional sobre un único *socket*<sup>7</sup> *Transmission*

---

<sup>4</sup> <http://thinkcorrectly.wordpress.com/2009/06/28/introduccion-a-ajax/>

<sup>5</sup> <http://thinkcorrectly.wordpress.com/2009/06/28/introduccion-a-ajax/>

<sup>6</sup> <http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-17>

<sup>7</sup> Abstracción software que funciona como punto final de las comunicaciones entre computadoras (puerta).

*Control Protocol* (TCP<sup>8</sup>). Lo cual permite al servidor interpretar parte de la petición de negociación como HTTP y entonces cambiar a WebSocket.

Muchas empresas y desarrolladores están apostando a esta nueva tecnología por los beneficios que reporta en cuanto a rendimiento de las aplicaciones. El protocolo WebSocket se ha convertido en un nuevo estándar para muchos marcos de trabajo y servidores. La pasarela Websockets de Kaazing<sup>9</sup>, Jetty WebSocket Servlet<sup>10</sup>, Socket.IO<sup>11</sup>, django-websocket<sup>12</sup> del proyecto Python y jWebSocket<sup>13</sup> son algunos de los muchos servidores que lo soportan.

La forma en que estos servidores logran comunicarse con las aplicaciones cliente es mediante una librería de clases. Esta varía en dependencia del lenguaje en que sea desarrollado el cliente, pero su funcionamiento es el mismo. A este conjunto de clases o procedimientos que permiten o facilitan la comunicación con el servidor se le conoce comúnmente como librería cliente (*Client-Library*).

Resulta de suma importancia para las tecnologías usadas en el desarrollo de aplicaciones cliente-servidor, contar con la mayor cantidad de librerías cliente posibles, ya que esto intensifica sus posibilidades de uso y éxito en el mercado. Las librerías cliente brindan interfaces de programación bien definidas que rigen comportamientos y estándares para la comunicación con el servidor.

jWebSocket es un marco de trabajo que utiliza el protocolo WebSocket para desarrollar aplicaciones en tiempo real para la Web. Este permite implementar las aplicaciones del lado del servidor empleando el lenguaje de programación multiplataforma Java. Mediante este protocolo es posible brindar una arquitectura muy flexible, a través de la cual las aplicaciones del lado del cliente pueden ser desarrolladas en cualquier lenguaje de programación. En la actualidad este marco

---

<sup>8</sup><http://es.kioskea.net/contents/internet/tcp.php3>

<sup>9</sup><http://kaazing.com/>

<sup>10</sup><http://www.eclipse.org/jetty/>

<sup>11</sup><http://socket.io/>

<sup>12</sup><http://pypi.python.org/pypi/django-websocket>

<sup>13</sup><https://jwebsocket.org/>

de trabajo sólo ofrece soporte para aquellas aplicaciones cliente que estén desarrolladas con los lenguajes Java o JavaScript.

Existen muchos lenguajes de programación y entornos de desarrollo en los cuales sería factible la implementación de una librería cliente para jWebSocket. Una buena opción sería el lenguaje C Sharp, el cual ocupa un lugar importante en el ranking mundial según la compañía de código estándar TIOBE<sup>14</sup>. C Sharp es el lenguaje más utilizado por la comunidad de desarrolladores gracias a su robustez, modernidad, sencillez de uso y su completa integración con la plataforma .NET que está clasificada como un entorno muy potente, robusto y con altísimas prestaciones.

Atendiendo a los elementos anteriores, se describe la siguiente **situación problemática**:

Hoy en día los desarrolladores del lenguaje de programación C Sharp no pueden aprovechar las potencialidades de jWebSocket porque este no brinda soporte para una librería cliente en este lenguaje. Esto obliga a los desarrolladores a invertir tiempo y recursos en implementar el protocolo WebSocket para lograr la conexión y transferencia de datos a bajo nivel con el servidor, alejándose de la lógica de la aplicación. Este hecho provoca una disminución de los niveles de productividad de un equipo de desarrollo a la hora de implementar soluciones jWebSocket con C Sharp. Además implementar el protocolo a bajo nivel de manera directa sin altos niveles de abstracción introduce un mayor número de errores a las aplicaciones. Lo que conlleva a un aumento del esfuerzo y el tiempo para corregirlos al mismo tiempo que disminuye la confiabilidad de las mismas.

De la problemática existente surge la siguiente interrogante que nos define el **problema científico**:

¿Cómo mejorar la productividad y confiabilidad de las aplicaciones clientes C Sharp para el marco de trabajo jWebSocket?

---

<sup>14</sup><http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

Por tanto para el desarrollo de la presente investigación se define como **objeto de estudio** comunicación mediante el protocolo WebSocket para aplicaciones clientes en C Sharp delimitándose como **campo de acción** comunicación mediante el protocolo WebSocket en aplicaciones clientes C Sharp para el marco de trabajo jWebSocket.

Para dar solución a esta problemática se determinó como **objetivo general**:

Desarrollar una librería que garantice mejorar la productividad y confiabilidad de las aplicaciones clientes C Sharp para el marco de trabajo jWebSocket.

Para dar cumplimiento al objetivo general se definen las siguientes **preguntas científicas**:

- ✓ ¿Cuáles son los fundamentos teórico-metodológicos de la comunicación mediante el protocolo WebSocket en aplicaciones clientes C Sharp?
- ✓ ¿Cuál es la situación actual de la comunicación mediante el protocolo WebSocket en aplicaciones clientes C Sharp?
- ✓ ¿Cómo desarrollar una librería que garantice mejorar la productividad y confiabilidad de las aplicaciones clientes C Sharp para el marco de trabajo jWebSocket.
- ✓ ¿Cómo validar la capacidad de la librería desarrollada para garantizar mayores niveles de la productividad y confiabilidad de las aplicaciones clientes C Sharp para el marco de trabajo jWebSocket?

Para agilizar y organizar el presente trabajo se definieron las siguientes **tareas de la investigación**:

- ✓ Fundamentación teórico-metodológica de la comunicación mediante el protocolo WebSocket en aplicaciones clientes C Sharp.
- ✓ Análisis de la situación actual de la comunicación mediante el protocolo WebSocket en aplicaciones clientes C Sharp.

- ✓ Desarrollo de una librería que garantice mejorar la productividad y confiabilidad de las aplicaciones clientes C Sharp para el marco de trabajo jWebSocket
- ✓ Comprobación de la capacidad de la librería desarrollada para garantizar mayores niveles de la productividad y confiabilidad en las aplicaciones clientes C Sharp para el marco de trabajo jWebSocket

Se definen las siguientes **variables de la investigación**:

**Independiente:** Librería desarrollada para aplicaciones clientes C Sharp para el marco de trabajo jWebSocket.

**Dependiente:**

- ✓ Grado de productividad en el desarrollo de aplicaciones clientes C Sharp para el marco de trabajo jWebSocket.
- ✓ Grado de confiabilidad de las aplicaciones clientes C Sharp para el marco de trabajo jWebSocket.

Para lograr un mejor entendimiento y obtener una mayor comprensión de lo que está sucediendo, se pondrán en práctica los siguientes **métodos científicos**:

**Métodos teóricos:**

- ✓ **Analítico-Sintético:** Este método permite analizar toda la teoría recopilada a través de los diferentes medios bibliográficos, documentos, libros, artículos, etc. Permitiendo el procesamiento de la información para arribar a diferentes conclusiones prácticas y teóricas a cerca del trabajo.
- ✓ **Histórico-Lógico:** Permite analizar la trayectoria completa del proceso de administración de aplicaciones distribuidas, así como el estudio histórico del mismo que permite ver deficiencias y proponer soluciones acorde a las necesidades.
- ✓ **Modelación:** Este método permite realizar una representación de la situación que se analiza. Permite obtener mediante diagramas y objetos una

mayor comprensión del problema y desarrollar un modelo para la aplicación a desarrollar a partir de la situación problemática.

- ✓ **Análisis Documental:** Mediante este método se hizo un estudio de una variada documentación referente a las librerías clientes y las herramientas utilizadas actualmente para lograrlo, con el objetivo de obtener a través de estas el análisis, las experiencias y las sugerencias que pudieran ser incorporadas a la tesis. Con el uso de este método se estudió además la documentación referente a las potenciales tecnologías a utilizar para desarrollar dicha librería, así como las experiencias de otros desarrolladores en el trabajo con las mismas.

Con este trabajo de diploma se pretende alcanzar los **posibles resultados:**

- ✓ Informe detallado con toda la base teórico-práctico sobre la cual se sustenta la solución propuesta.
- ✓ Librería desarrollada para aplicaciones clientes C Sharp para el marco de trabajo jWebSocket.

El presente trabajo está estructurado de la siguiente manera: Introducción, tres capítulos de contenido, Conclusiones, Recomendaciones, Bibliografía, Referencias Bibliográficas y Glosario de Términos

A continuación se hace una breve descripción del contenido de los capítulos:

**Capítulo1:** Fundamentación Teórica: Se realiza la fundamentación teórica de la investigación. Se expone un estudio del estado del arte sobre la abstracción de lenguajes de programación existentes en la actualidad.

**Capítulo 2:** Características, Análisis y Diseño del Sistema: Brinda una fundamentación de la solución propuesta, a partir de la cual se describen las actividades de análisis de la solución, seguidas por la descripción de los procesos del sistema y de las etapas de diseño e implementación que conllevan a la obtención del software.



**Capítulo 3:** Implementación y Validación del Sistema: Se elaboran y documentan las pruebas realizadas a la solución propuesta para demostrar el cumplimiento de los requerimientos de la misma. Se realiza un análisis de los resultados de la aplicación en un entorno real, comparando indicadores antes y luego de la solución.

## Introducción

En el presente capítulo se elabora la base teórica que sustenta y respalda la presente investigación. Con este objetivo se exponen los principales conceptos asociados al dominio del problema permitiendo una mejor comprensión del objeto de estudio y campo de acción en cuestión. Se realiza además un análisis de soluciones existentes con características similares y una panorámica general acerca de la comunicación en tiempo real con WebSocket y en específico para las aplicaciones clientes C Sharp. Así como un estudio de las metodologías, tecnologías y herramientas a tener en cuenta para el desarrollo de la solución propuesta.

### 1.1 Conceptos Asociados al Dominio del Problema

Los conceptos fundamentales que orientaron la presente investigación se refieren a la comunicación mediante el protocolo WebSocket, jWebSocket y las librerías cliente.

Durante años se han desarrollado aplicaciones web utilizando el protocolo HTTP que fue intencionalmente diseñado como un simple protocolo de solicitud-respuesta. Este esquema no proporciona una comunicación bidireccional entre el cliente y el servidor, por lo que es necesario recurrir a diversas técnicas como *polling* y *long polling* para lograr simular una transferencia de datos en tiempo real.

Con el objetivo de solucionar el problema de la transferencia de datos en tiempo real surge el protocolo WebSocket que permite realizar conexiones bidireccionales entre un cliente y un servidor. Consiste en un mecanismo de handshake<sup>15</sup> seguido de intercambios de mensajes sobre la capa TCP. 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)

---

<sup>15</sup><http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-10>

Esta definición se basa en la necesidad de establecer una comunicación bidireccional mediante el protocolo TCP entre un navegador Web y un servidor de aplicaciones. Otros autores basan sus conceptos en la capacidad de disminución del ancho de banda que se logra con la reducción de las cabeceras HTTP al utilizar el protocolo WebSocket como medio de comunicación. PetterLubbers lo plantea con este enfoque en la siguiente definición.

WebSoket es una tecnología que proporciona un canal de comunicación bidireccional y *full-duplex* sobre un único socket TCP. Soluciona las limitaciones del protocolo HTTP, al establecer una comunicación full-duplex TCP entre el cliente y el servidor, sustituyendo la comunicación half-duplex HTTP. Se reduce, en grandes proporciones, el tráfico en la red teniendo en cuenta que al establecer la comunicación WebSocket entre el cliente y el servidor solo hay un envío de 4 bytes, eliminando las cabeceras HTTP. (HTML5 Websockets and communication, 2010).

Por otra parte la IETF da un enfoque más general respecto a la comunicación utilizando el protocolo WebSocket como se muestra a continuación.

WebSocket permite la comunicación bidireccional entre un cliente que ejecuta código no confiable en un ambiente controlado y un anfitrión remoto que ha optado por comunicaciones desde ese código. El modelo de seguridad utilizado para ello es el modelo de seguridad basado en el origen comúnmente utilizado por los navegadores web. Este protocolo consiste en un apretón de manos de apertura, seguida por la estructuración en capas del mensaje básico a través de TCP. El objetivo de esta tecnología es proporcionar un mecanismo para aplicaciones basadas en navegadores que necesitan una comunicación bidireccional con los servidores que no dependen de la apertura de múltiples conexiones HTTP. (Fette, y otros, 2011)

Dadas las características de la presente investigación y teniendo en cuenta las definiciones mencionadas anteriormente, se asume que WebSocket no es más que una tecnología que permite realizar una comunicación bidireccional sobre un canal TCP entre una aplicación cliente y un servidor. Dicha comunicación debe ser

iniciada por el cliente al cual le responderá el servidor. Cuando la conexión se establece se realiza una actualización del protocolo HTTP al protocolo TCP, permitiendo el envío de información en ambas direcciones sin necesidad de realizarse solicitudes por parte del cliente.

Con el surgimiento del protocolo WebSocket se han creado diversos marcos de trabajo para la comunicación cliente – servidor. jWebSocket es un servidor de aplicaciones y a la vez un marco de trabajo creado específicamente para soportar el desarrollo de aplicaciones web en tiempo real utilizando el protocolo WebSocket. (Framework Approach for WebSockets, 2011)

Se puede establecer la comunicación con dicho marco de trabajo utilizando una librería cliente que no es más que una interfaz de programación de aplicaciones (API) que se utiliza para escribir aplicaciones cliente. Proporciona la creación de bloques genéricos para la construcción de aplicaciones cliente. (Sybase, 2009). La implementación del protocolo WebSocket del lado del cliente en forma de librería hace que la misma sea más reutilizable y aproveche todos los beneficios que brinda este tipo de arquitectura.

Aunque esta definición plantea correctamente lo que es una librería, la misma no especifica el tipo de aplicaciones clientes en las que se puede utilizar. Es válido aclarar que una librería cliente implementada en el lenguaje de programación C Sharp podría ser utilizada por cualquier tipo de solución que se desarrolle con este lenguaje.

## 1.2 Análisis de Soluciones Existentes

En la actualidad existen toda una serie de servidores y marcos de trabajo que utilizan la tecnología WebSocket en soluciones para diferentes plataformas y lenguajes de programación. Algunos de los más significativos y con mejor aceptación por la comunidad de desarrolladores son la pasarela WebSocket de Kaazing, Jetty WebSocket Servlet, Socket.IO, django-websocket del proyecto Python y jWebSocket por mencionar algunos.

La mayoría de estos servidores utilizan del lado del cliente algún tipo de librería que implemente el protocolo WebSocket para realizar la conexión y transferencia de datos con el servidor. De los marcos de trabajo mencionados anteriormente se estudiaron con profundidad `django-websocket` del proyecto Python y `jWebSocket` porque ambos poseen una implementación de librerías que pueden servir como guía para desarrollar una solución similar.

Para el caso de `django-WebSocket` posee una librería cliente en Python que se encarga de implementar la API WebSocket y gestionar la lógica de la aplicación con dicho servidor. Pero como Python es un lenguaje estructurado y la librería cliente C Sharp que se desea implementar utiliza un lenguaje orientado a objetos, no es de mucha utilidad centrarnos en el estudio de esta librería porque la arquitectura utilizada y la forma de reutilizar el código no la podemos aplicar en la solución que se desea desarrollar.

Por otra parte tenemos el marco de trabajo `jWebSocket` que le da soporte a librerías cliente escritas en el lenguaje de programación JavaScript y Java:

- ✓ Cliente JavaScript WebSocket API: Está compuesto por un conjunto de funcionalidades que se encarga de gestionar la lógica de la aplicación para establecer la comunicación con un servidor `jWebSocket`. Utilizando la implementación nativa del protocolo WebSocket disponible para los navegadores web. ([jwebsocket.org](http://jwebsocket.org), 2011)
- ✓ Cliente java: El cliente de Java en una red WebSocket realiza exactamente las mismas tareas que un cliente Web y por lo tanto en el nivel más bajo proporciona los mismos métodos y eventos disponibles en la API WebSocket. Todos los mecanismos de comunicación que implementa son igualmente aplicables a los clientes de Java que se ejecutan en Android. ([jwebsocket.org](http://jwebsocket.org), 2011)

Esta librería cliente Java se toma como fuerte guía y referencia a la hora de implementar la librería cliente C Sharp, puesto que ambas utilizan lenguajes de

programación muy similares, lo que facilita la comprensión del código estudiado y además tienen en común el marco de trabajo `WebSocket`. Aquí se puede estudiar y ver el mecanismo de abstracción que es necesario realizar para implementar un objeto que sea común del lado del cliente y del servidor.

En la versión 4.5 del *Framework*<sup>16</sup> .NET se da a conocer la implementación de la clase `WebSocket`. Esta implementación se encuentra en el espacio de nombre `System.Net` y todavía está sujeta a cambios. Utilizar este componente en el desarrollo de la librería cliente C Sharp traería grandes ventajas porque ahorraría gran parte de la implementación del protocolo `WebSocket`. El problema es que .NET es software propietario y no se brinda el código fuente de esta clase para someterla a cambios que serían necesarios para la solución propuesta. También hay que tener en cuenta que como es una primera versión y está sometida a futuros cambios esto podría no ser muy beneficioso, así que se decide no utilizarla e implementar el protocolo `WebSocket` desde cero. (msdn, 2011)

De manera general todos estos servidores y marcos de trabajo estudiados se pueden utilizar de una forma u otra como guía, pero no cumplen con todos los requisitos que necesita la librería cliente C Sharp. Por esto se hace necesario iniciar una investigación que logre diseñar e implementar dicha librería con el objetivo de lograr conexión y transferencia de datos en tiempo real con el marco de trabajo `WebSocket`.

### 1.3 Metodología usada para el desarrollo de la solución

Todo proceso de desarrollo de una aplicación informática debe estar regido y orientado por una metodología de desarrollo de software, que guíe los procesos y permita tener un registro detallado del avance de la investigación. Las metodologías pueden ser robustas o ágiles. Las metodologías robustas o pesadas están concebidas para guiar el proceso de desarrollo de los software de gran envergadura, cuando un proyecto requiere de gran cantidad de documentación,

---

<sup>16</sup> Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos

vaya a ser realizado en un tiempo considerablemente largo y existe la posibilidad de que pase por las manos de varios equipos de trabajo.

Por su parte las metodologías ágiles intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en los clientes y los resultados. Se basan en promover iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto, logrando que se minimicen los riesgos desarrollando software en cortos tiempo (Metodologías Tradicionales Vs. Metodologías Ágiles, 2011) Teniendo en cuenta los aspectos anteriores se realizó un análisis de la metodología robusta RUP y la ágil SXP para identificar cual es la más apropiada para el desarrollo de la librería cliente C Sharp.

## **Proceso Unificado de Desarrollo**

El Proceso Unificado de Rational (RUP) es un proceso formal que provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales, respetando cronograma y presupuesto. Es guiado por casos de uso y centrado en la arquitectura, iterativo e incremental y utiliza UML como lenguaje de notación. (Figueroa, y otros, 2011)

### **Consta de 4 fases principales:**

- ✓ **Inicio:** El objetivo en esta etapa es determinar la visión del proyecto.
- ✓ **Elaboración:** En esta etapa el objetivo es determinar la arquitectura óptima.
- ✓ **Construcción:** En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- ✓ **Transición:** El objetivo es llegar a obtener el despliegue del proyecto.

## **SXP**

SXP está compuesta por las metodologías SCRUM y XP, ofreciendo una estrategia metodológica, a partir de la introducción de procedimientos ágiles que permitan

actualizar los procesos de software para el mejoramiento de la actividad productiva. Esta metodología fomenta el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo.

## Consta de 4 fases principales:

- ✓ **Planificación-Definición:** Donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
- ✓ **Desarrollo:** Es donde se realiza la implementación del sistema hasta que esté listo para ser entregado;
- ✓ **Entrega:** Es la puesta en marcha; y por último.
- ✓ **Mantenimiento:** Es la fase donde se realiza el soporte para el cliente.

De cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la Lista de Reserva del Producto, definición de las Historias de Usuario, Diseño, Implementación, Pruebas, entre otras; de donde se generan artefactos para documentar todo el proceso. Las entregas son frecuentes, y existe una refactorización continua, lo que permite mejorar el diseño cada vez que se le añade una nueva funcionalidad.

SXP está especialmente indicada para proyectos con pequeños equipos de trabajo, un constante cambio de requisitos o requisitos imprecisos, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Fomenta el trabajo en equipo, con un objetivo claro, permitiendo el seguimiento y control de las tareas a realizar. (SXP, Metodología Ágil para el Desarrollo de Software, 2010)

No existe una metodología universal que indique cómo crear todo tipo de software. Cuando un equipo decide construir un producto, se debe escoger la metodología a utilizar teniendo en cuenta las características, complejidad, envergadura del proyecto y el tipo de contrato establecido para el mismo. (SXP, Metodología Ágil para el Desarrollo de Software, 2010). Debido a las ventajas que proporciona la



metodología SXP y por ser la que más se ajusta a las necesidades del proyecto, se escoge la misma para guiar el desarrollo de la librería cliente C Sharp.

El desarrollo de software es un proceso que consta de diferentes actividades, entre las que se destacan: el modelado del negocio, el desarrollo de requisitos, el diseño arquitectónico, el diseño detallado, la programación y las pruebas de la aplicación. En cada una de estas actividades se debe elaborar diferentes tipos de modelos utilizando algún lenguaje de modelado de software. En el siguiente epígrafe se realiza un análisis del lenguaje a utilizar para el modelado de la librería cliente C Sharp.

## 1.4 Lenguaje de Modelado de Software

El Lenguaje de Modelado Unificado (UML) y la notación *Business Process Modeling Notation* (BPMN) son los dos lenguajes más utilizados en la Industria de Software para elaborar estos modelos, que son indispensables para analizar, diseñar, programar y probar una aplicación. A continuación se muestra un detallado análisis del lenguaje UML.

Lenguaje Modelado Unificado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. Por lo anterior expuesto se decide utilizar UML como lenguaje de modelado para la librería cliente C Sharp. En el siguiente epígrafe se realiza un estudio sobre las herramientas

CASE que soportan el lenguaje UML para utilizarla en el desarrollo de esta solución.

## 1.5 Herramienta CASE

Las herramientas *Computer Aided Software Engineering*, Ingeniería de Software Asistida por Ordenador (CASE) propician un conjunto de métodos y técnicas automatizadas que brindan ayuda y dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todo el ciclo de vida del desarrollo de un software, reduciendo el esfuerzo, el costo y el tiempo.

Dichas herramientas se encuentran en una continua evolución, por lo que existe una gran variedad de proveedores y productos, cada uno de ellos con diferentes aplicaciones y especificaciones. A continuación se hace una caracterización de algunas de ellas que permitirá adquirir los elementos necesarios para determinar cuál es la más idónea para especificar y diseñar la solución propuesta. (Herramientas Case, 2011)

### Visual Paradigm

Visual Paradigm es una poderosa herramienta CASE que hace uso del lenguaje modelado unificado (UML) con soporte multiplataforma, que proporciona un ciclo de vida completo del desarrollo de software, excelentes facilidades de interoperabilidad con otras aplicaciones, compatibilidad entre versiones, así como dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Dicha herramienta brinda una serie de facilidades que se mencionan a continuación: (Visual Paradigm)

- ✓ Soporta un conjunto de estándares entre los que se encuentran UML, SysML, BPMN, XML y XMI.
- ✓ Soporte de modelado UML, modelado de procesos de negocios y un generador de mapeo de objetos-relacionales para los lenguajes de programación Java, .NET y PHP.

- ✓ Ofrece herramientas para la generación de reportes en formatos html, pdf y doc.
- ✓ Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.
- ✓ Interoperabilidad e integración. Permite la integración con un conjunto de herramientas (Visio drawing, Rational Rose, ERwin Data Modeler Project, Microsoft Excel y Microsoft Word document) e intercambiar diagramas UML y modelos usando representaciones industriales comunes.
- ✓ Modelado de base de datos. Proporciona una mayor documentación de la base de datos y diagramas de mapeo de relación de objetos.
- ✓ Integración con herramientas para el control de versiones.
- ✓ Diseño de prototipo de Interfaz de Usuario. Permite insertar información adicional a los diagramas mediante notas y comentarios para describir sus elementos lo que facilita la revisión de los prototipos así como el trabajo en equipo.

## Rational Rose

Rational Rose es una herramienta CASE que da soporte al modelado visual con UML cubriendo todo el ciclo de vida de un proyecto. Es compatible con la metodología RUP que permite crear los diagramas que se van generando durante el proceso de ingeniería en el desarrollo del software. Se enmarca dentro del desarrollo de modelado para fines académicos, investigativos y comerciales. Además brinda una serie de facilidades que se mencionan a continuación: ()

- ✓ Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en "DesignPatterns: Elements of Reusable Object-Oriented Software".
- ✓ La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables.
- ✓ Capacidad de análisis de calidad de código.
- ✓ El Add-In para modelado Web provee visualización, modelado y las herramientas para desarrollar aplicaciones Web.

- ✓ Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- ✓ Capacidad para integrarse con sistemas de control de versiones.
- ✓ Publicación web y generación de informes para optimizar la comunicación dentro del equipo.
- ✓ Sistemas operativos y plataformas de hardware apropiadas: Windows 2000, Windows NT, Windows XP.

De las herramientas CASE analizadas anteriormente para realizar el modelado de software, se decide seleccionar Visual Paradigm ya que la universidad cuenta con una licencia comercial de la misma.

## 1.6 Herramientas y Tecnologías Utilizadas

En el siguiente epígrafe se realiza un análisis del Entorno de Desarrollo Integrado más adecuado para el desarrollo de la librería cliente C Sharp.

### 1.6.1 Entorno integrado de desarrollo – IDE

#### Microsoft Visual Studio

Microsoft Visual Studio es un IDE de gran alcance que asegura la calidad del código en todo el ciclo de vida de la aplicación, desde el diseño hasta la implementación. Tanto si estás en el desarrollo de aplicaciones para SharePoint, Internet, Windows, Windows Phone y más allá, Visual Studio es un IDE para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, C Sharp, Visual J Sharp, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (Microsoft, 2011)

#### MonoDevelop

MonoDevelop es un entorno de desarrollo integrado libre y gratuito, diseñado

primordialmente para C Sharp y otros lenguajes .NET como Nemerle, Boo, Java (vía IKVM.NET) y en su versión 2.2 Python. MonoDevelop originalmente fue una adaptación de SharpDevelop para Gtk#, pero desde entonces se ha desarrollado para las necesidades de los desarrolladores del Proyecto Mono. El IDE incluye manejo de clases, ayuda incorporada y completamiento de código. MonoDevelop ya cuenta con soporte completo para GNU/Linux, Windows y Mac, completando así un hito para ser un verdadero IDE Multiplataforma. (Monodevelop, 2011)

Las características anteriormente expuestas sobre Visual Studio y MonoDevelop demuestran las potencialidades de ambos para el desarrollo de aplicaciones en C Sharp. Sin embargo, para el desarrollo de la librería cliente C Sharp se selecciona como Entorno Integrado de Desarrollo a Visual Studio, debido a que se tiene una mayor experiencia y familiarización con esta herramienta. Además Microsoft Visual Studio 2010 Ultimate incluye potentes herramientas que simplifican todo el proceso de desarrollo de aplicaciones, de principio a fin. Los equipos pueden observar una mayor productividad y ahorro de costes al utilizar características de colaboración avanzadas, así como herramientas de pruebas y depuración integradas que le ayudarán a crear siempre un código de gran calidad. (Microsoft, 2011)

## 1.6.2 Marco de Trabajo

El Framework de .Net es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Los principales componentes de este entorno son:

- ✓ Lenguajes de compilación
- ✓ Biblioteca de clases de .Net
- ✓ CLR (Common Language Runtime)

.Net Framework soporta múltiples lenguajes de programación y aunque cada lenguaje tiene sus características propias, es posible desarrollar cualquier tipo de

aplicación con cualquiera de estos lenguajes. Existen más de 30 lenguajes adaptados a .Net, desde los más conocidos como C Sharp, Visual Basic o C++ hasta otros lenguajes menos conocidos como Perl o Cobol.

## **Common Language Runtime (CLR)**

El CLR es el verdadero núcleo del Framework de .Net, ya que es el entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios que ofrece el sistema operativo estándar Win32.

## **Biblioteca de clases de .Net**

Cuando se está programando una aplicación muchas veces se necesitan realizar acciones como manipulación de archivos, acceso a datos, conocer el estado del sistema, implementar seguridad, etc. El Framework organiza toda la funcionalidad del sistema operativo en un espacio de nombres jerárquico de forma que a la hora de programar resulta bastante sencillo encontrar lo que se necesita.

## **Ensamblados**

Uno de los mayores problemas de las aplicaciones actuales es que en muchos casos tienen que tratar con diferentes archivos binarios (DLL's), elementos de registro, conectividad abierta a bases de datos (ODBC), etc.

Para solucionarlo el Framework de .Net maneja un nuevo concepto denominado ensamblado. Los ensamblados son ficheros con forma de EXE o DLL que contienen toda la funcionalidad de la aplicación de forma encapsulada. Por tanto la solución al problema puede ser tan fácil como copiar todos los ensamblados en el directorio de la aplicación.

### **1.6.3 Herramientas de Control de Versiones**

Para el desarrollo de un proyecto de software de esta envergadura se hace indispensable la utilización de una herramienta para el control de versiones debido a las necesidades de controlar los cambios realizados al código fuente.

Un sistema de control de versiones es un sistema de gestión de archivos y directorios, cuya principal característica es mantener el historial de cambios y modificaciones que se han realizado sobre dichos archivos a lo largo del tiempo. Es importante decir que estos sistemas no solo se limitan a gestionar archivos de texto sino que también gestionan documentos, imágenes y ficheros de todo tipo. Los sistemas de control de versiones utilizados a nivel internacional con gran aceptación y popularidad guardan toda la información en un repositorio central accesible a través de la red, permitiendo el trabajo colaborativo entre varios puestos de trabajo y a su vez proporcionando una mayor seguridad y disponibilidad de los datos. A continuación se hace una caracterización de algunos sistemas de control de versiones con el objetivo de determinar el más idóneo para garantizar mayor seguridad y disponibilidad de los datos.

## **Subversion**

Subversion es un sistema de control de versiones completamente equipado que fue originalmente diseñado para reemplazar a CVS. Desde entonces se ha expandido más allá de su objetivo original, pero su modelo básico, el diseño y la interfaz fueron fuertemente influenciados por CVS por lo que debido a estas particularidades los usuarios de CVS se sienten muy cómodos al interactuar con Subversion. (Apache Software Foundation, 2011)

Este sistema presenta varias características importantes las cuales aparecen a continuación.

- ✓ Los directorios son versionados.
- ✓ Resolución de conflictos de forma interactiva.
- ✓ Gestiona de manera eficaz los archivos binarios.
- ✓ Bloqueo de archivos.
- ✓ Vinculaciones para lenguajes de programación.
- ✓ Vinculación de varios repositorios.
- ✓ Soporte para desarrolladores.
- ✓ Desarrollo Paralelo.

Dentro de las características planteadas anteriormente se profundiza en varias de ellas debido a la relevancia de las mismas.

## **Soporte para desarrolladores**

Los desarrolladores tienen acceso a todas las funcionalidades de Subversion desde diferentes entornos integrados de desarrollo (IDEs) como Eclipse y NetBeans, entre otros.

## **Desarrollo Paralelo**

Subversion permite el desarrollo paralelo, para que miembros individuales del equipo de trabajo puedan completar las diferentes partes y versiones de un proyecto al mismo tiempo, sin tener la necesidad de esperar por que otro compañero termine las tareas que está realizando.

## **Git**

Git es un sistema de control de versiones, diseñado para el trabajo con proyectos de cualquier tamaño con gran rapidez y eficiencia. Tiene una forma diferente y revolucionaria en su sistema de guardado haciendo que cada copia de trabajo sea un repositorio en sí mismo y contenga todo el historial de modificaciones. Esta es una característica que lo hace muy eficiente porque se puede disponer de toda la información necesaria para trabajar sin conexión y sincronizar los cambios una vez restablecida la conexión.

## **Entre sus características se encuentran:**

- ✓ Eficiencia: Git es un sistema de control de versiones que fue desarrollado para ser eficiente y como resultado se obtiene un sistema altamente eficiente y veloz en el que duplicar un repositorio lleva escasos segundos.
- ✓ Desarrollo no lineal: Múltiples ramas paralelas que se dividen y se unen en distintos puntos es una de las características que brinda este sistema que evita la separación del código en ramas de trabajo que pueden o no incorporarse al código base.



- ✓ Limpieza del espacio de trabajo: Git a diferencia de los otros sistemas de control de versiones únicamente crea un fichero oculto en la raíz de la copia de trabajo en la que se almacena toda su información sobre versiones.
- ✓ Desaparición del número de versión del repositorio, aunque dispone de algunos sistemas para emularlo como un “descriptor” de versión.
- ✓ Git, a diferencia de otros sistemas de control de versiones, almacena el fichero completo en lugar de diferencias respecto al anterior.
- ✓ Autenticación criptográfica del repositorio.

Teniendo en cuenta las características expuestas, se selecciona Subversion como herramienta de control de versiones para ser utilizado en el desarrollo de la librería cliente C Sharp, ya que el equipo de desarrollo tiene gran experiencia en el trabajo con esta herramienta y presenta una gran facilidad de uso

## 1.6.4 Herramientas de Control de Versiones del lado del cliente

Los clientes permiten la gestión de cambios que se realizan sobre los elementos de algún producto. Permitiendo una conexión entre él como cliente y el servidor, para un mejor manejo de los archivos locales.

### TortoiseSVN

Es un cliente gratuito de código abierto para el sistema de control de versiones Subversion. TortoiseSVN maneja ficheros y directorios a lo largo del tiempo. Los ficheros se almacenan en un repositorio central. El repositorio es prácticamente lo mismo que un servidor de ficheros ordinario, salvo que recuerda todos los cambios que se hayan hecho a sus ficheros y directorios. Esto permite que pueda recuperar versiones antiguas de sus ficheros y examinar la historia de cuándo y cómo cambiaron sus datos, y quién hizo el cambio. (team, 2011)

### Características:

- ✓ Integración con el Shell de Windows: TortoiseSVN se integra perfectamente en el Shell de Windows (por ejemplo, el explorador). Esto significa que puede seguir trabajando con las herramientas que ya se conocen.

- ✓ No está obligado a usar el explorador de Windows. Los menús contextuales de TortoiseSVN también funcionan en otros administradores de archivos y en el diálogo Fichero/Abrir que es común a la mayoría de aplicaciones estándar de Windows. Sin embargo, debe tener en cuenta que TortoiseSVN está desarrollado con la mirada puesta en hacerle extensión del Explorador de Windows. Por este motivo, puede que en otras aplicaciones la integración no sea tan completa y que, por ejemplo, los íconos sobreimpresionados en las carpetas no se muestren.
- ✓ Los Iconos sobreimpresionados: El estado de cada carpeta y fichero versionado se indica por pequeños íconos sobreimpresionados. De esta forma, se puede ver fácilmente el estado en el que se encuentra su copia de trabajo.
- ✓ Fácil acceso a los comandos de Subversion: Todos los comandos de Subversion están disponibles desde el menú contextual del explorador. TortoiseSVN añade su propio submenú allí.

## RapidSVN

Es una plataforma de interfaz gráfica de usuario, para el sistema de revisión de Subversion. Este proyecto también incluye un cliente de Subversion C++ API. RapidSVN está licenciado bajo la v3 de GNU General Public License.

Utiliza las mejores características de los clientes de otras arquitecturas de control de versiones. Si bien es bastante fácil para los nuevos usuarios de Subversion trabajar con él, también debe ser lo suficientemente potente como para que los usuarios con experiencia sean aún más productivos. (RapidSVN, 2011)

### Características:

- ✓ Simple: Proporciona una interfaz fácil de usar para las características de Subversion.
- ✓ Eficiente: Simple para los principiantes pero lo suficientemente flexible como para aumentar la productividad para los usuarios de Subversion.

- ✓ Portátil: Se ejecuta en cualquier plataforma: Linux, Windows, Mac OS / X, Solaris.
- ✓ Rápido: Completamente escrito en C + +.

Se analizaron algunas de las herramientas como por ejemplo Tortoise y Rapsdsvn, y se decide utilizar Tortoise debido a todas las ventajas y funcionalidades que ofrece además de tener una completa Integración con el Shell de Windows.

## Conclusiones

En el capítulo se realizó un estudio de las tecnologías y tendencias actuales en cuanto al desarrollo de aplicaciones en tiempo real para la web y los marcos de trabajo que permiten el desarrollo de este tipo de aplicaciones. Se analizaron y se eligieron las principales herramientas, metodologías de desarrollo y marcos de trabajo a utilizar. Con el objetivo de desarrollar una librería cliente que implemente el protocolo WebSocket, mediante la cual se puedan desarrollar aplicaciones cliente para el marco de trabajo jWebSocket.

Teniendo en cuenta el análisis realizado previamente, se toma como decisión la implementación de dicha librería cliente en el lenguaje de programación C Sharp. La misma se encargara de gestionar toda la lógica de la aplicación así como lograr la integración del marco de trabajo jWebSocket con aplicaciones cliente desarrolladas en la plataforma .NET. Logrando agilizar el proceso de desarrollo de software y prevenir errores humanos y de implementación del protocolo WebSocket.

## Introducción

En el presente capítulo se describen las funcionalidades del sistema, así como sus características principales. Se detallan y explican brevemente los artefactos que posee la metodología de desarrollo SXP propuesta en el capítulo anterior. Para una mejor comprensión del contexto del sistema se define el modelo de negocio y se describen sus requisitos funcionales y no funcionales, las historias de usuario y las tareas de ingeniería asociadas a las mismas.

### 2.1 Propuesta de Solución

Una librería que garantice mejorar la productividad y confiabilidad de las aplicaciones clientes C Sharp para el marco de trabajo jWebSocket, no es más que un conjunto de clases y funcionalidades que brindan una interfaz bien definida y estructurada para gestionar la comunicación entre una aplicación cliente y el marco de trabajo jWebSocket. La misma es de gran utilidad para la comunidad de desarrolladores que utilizan el lenguaje de programación C Sharp, porque al utilizar esta librería pueden centrarse en la lógica de la aplicación sin perder tiempo en la implementación del protocolo WebSocket para lograr la comunicación con dicho marco de trabajo.

Para el desarrollo de dicha librería se implementó el protocolo WebSocket con el propósito de establecer una conexión con el marco de trabajo jWebSocket o cualquier otro servidor que utilice este protocolo de comunicación. Con el objetivo de gestionar la transferencia de datos entre el marco de trabajo y la aplicación cliente desarrollada con esta librería se realizó una abstracción del modelo de datos que utiliza dicho marco de trabajo para lograr compatibilidad entre los tipos de datos de ambos.

Dentro de las ventajas que brinda la utilización de esta librería se encuentra poder desarrollar una aplicación cliente en tiempo real para el marco de trabajo jWebSocket que garantice mejorar la productividad y confiabilidad de la misma. Otra ventaja considerable es poder utilizar una interfaz de programación de

aplicaciones (API) bien definida y testeada, lo que genera seguridad y rapidez a la hora de desarrollar este tipo de aplicaciones.

## 2.2 Planificación del Proyecto por Roles

La metodología SXP define diferentes roles para repartir las responsabilidades y lograr un resultado exitoso en el proceso de desarrollo de software. De esta forma se le asigna a cada integrante del proyecto una responsabilidad con el objetivo de coordinar e integrar sus esfuerzos para lograr un objetivo común. A continuación en la (Tabla 1) se muestra la asignación de roles pertenecientes al proyecto, así como las principales responsabilidades de cada uno de estos.

Tabla 1: Planificación del proyecto por roles

Rol	Responsabilidad	Nombre
Líder del Proyecto	Debe asegurar que el proyecto se está llevando a cabo de acuerdo con las prácticas y que todo funciona según lo planeado. Su principal trabajo es remover impedimentos y reducir riesgos del producto. Coordina y facilita las reuniones. Asegura que se consiguen los objetivos de cada iteración.	Yamila Vigil Regalado
Gerente	Es el responsable de tomar las decisiones finales, acerca de estándares y convenciones a seguir durante el proyecto. Participa en la selección de objetivos y requerimientos. Tiene la responsabilidad de controlar el progreso y trabaja junto con el Jefe de Proyecto en la reducción de la Lista de Reserva del Producto.	Alexander Schulze
Especialista	Es necesario que conozca a fondo el proceso para el desarrollo de software. Es una especialización que está activa, el miembro del grupo de trabajo	Rebecca Schulze Eduardo Bourzac Hernández

# Capítulo 2. Características, Análisis y Diseño del Sistema

	que la desempeña siempre está ejecutándola y alcanzando un grado mayor de conocimientos en el tema, en este caso como Diseñador Gráfico.	
Consultor	Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas, además aportan ideas y experiencias para el beneficio del sistema en desarrollo. Esta es una especialización menos activa, quien la ejecuta funciona en este rol por un corto período de tiempo.	Alexander Schulze Rebecca Schulze
Cliente	Participa en las tareas que involucran la lista de reserva del producto.	Alexander Schulze
Programador	Elabora el código de las nuevas funcionalidades a implementar. Escribe las pruebas unitarias. Debe existir una comunicación y coordinación adecuada entre los programadores y el resto del equipo.	Rolando Betancourt Toucet
Analista	Escribe las historias de usuario y las pruebas funcionales para validar su implementación.	Rolando Betancourt Toucet
Diseñador	Encargado del diseño del sistema y de los prototipos de interfaces, son los máximos responsables de la realización del diseño de las metáforas y supervisan el proceso de construcción.	Rolando Betancourt Toucet
Encargado de Pruebas	Es el encargado de ayudar al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.	Rolando Betancourt Toucet

Arquitecto	Se vincula con el analista y el diseñador ya que su trabajo tiene que ver con la estructura y el diseño del sistema. Ayuda en el diseño de las metáforas.	Rolando Betancourt Toucet
------------	---	------------------------------

## 2.3 Modelo de Historias de Usuario del Negocio

Dentro de los artefactos que genera la metodología SXP se encuentra la plantilla del Modelo Historias de Usuario del Negocio, donde se definen las características específicas del negocio, así como la forma en que interactúa el sistema con los clientes y viceversa. En este se realiza la especificación de los usuarios y trabajadores que intervienen así como su interacción con las historias de usuario. Además se crea una descripción precisa de los elementos que intervienen en el negocio en cuestión.

Este Modelo de Historias de Usuario del Negocio se corresponde con los objetivos de la librería cliente C Sharp. En el mismo se describen los requerimientos principales asociados al funcionamiento que presenta dicha librería. Dicho modelo detalla el negocio que está presente en el uso de una librería como intermediario entre una aplicación cliente y el marco de trabajo jWebSocket.

A continuación se muestra en la (Fig. 1) el modelo de historias de usuario del negocio que se propone para la librería cliente C Sharp.

**Aplicación C Sharp:** En este caso la aplicación desarrollada con C Sharp es la única beneficiada de la implementación de una librería cliente en dicho lenguaje utilizando el protocolo WebSocket, donde la librería es el único sistema informático que se desea desarrollar. O sea es la aplicación el actor del negocio al cual están dirigidas todas las historias de usuario.

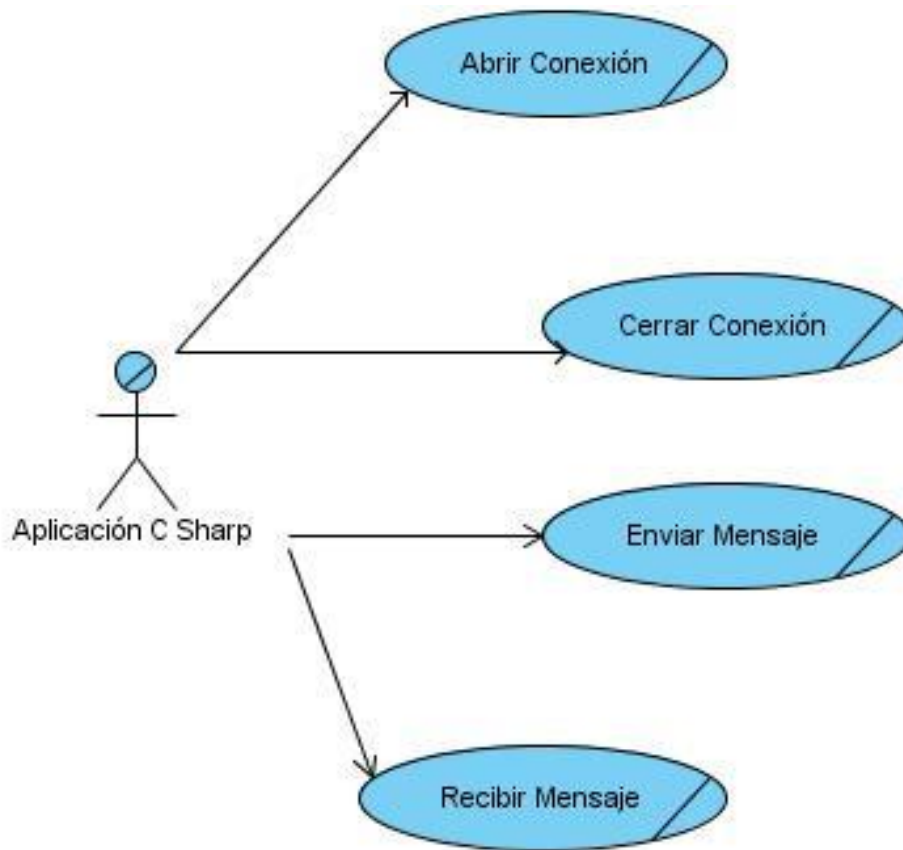


Fig. 1: Modelo de Historias de Usuario del Negocio

## 2.4 Lista de Reserva del Producto (LRP)

La lista de Reserva del Producto es una lista con prioridad en la cual quedan reflejadas todas las tareas a desarrollar en el proyecto durante el tiempo de vida del mismo. A medida que vayan aumentando los conocimientos relacionados con el producto y se vaya identificando con más claridad las necesidades del cliente, irá aumentando o variando esta lista, estos cambios solo pueden hacerse entre iteraciones.

A continuación se presenta la (Tabla 2) que muestra la lista priorizada contenida en el LRP perteneciente a la librería cliente C Sharp donde se muestra la relación existente entre la prioridad y el tiempo de realización asignado a cada tarea.



# Capítulo 2. Características, Análisis y Diseño del Sistema

Tabla 2: Lista de Reserva del Producto

Prioridad	Ítem *	Descripción	Estimación	Estimado por
<b>Muy Alta</b>				
	1	Crear el Handshake.	2 días	Analista
	2	Procesar el Handshake	2 días	Analista
	3	Establecer conexión.	3 días	Analista
<b>Alta</b>				
	4	Enviar token binario.	2 semanas	Analista
	5	Enviar token de texto.	2 semanas	Analista
	6	Enviar token fragmentado.	2 semanas	Analista
	7	Enviar paquete	1 semana	Analista
	8	Recibir token.	2 semanas	Analista
	9	Recibir e Interpretar fragmentos de token.	2 semanas	Analista
<b>Media</b>				
	10	Finalizar conexión.	1 semanas	Analista
<b>RNF (Requisitos No Funcionales)</b>				
	8	Garantizar la integridad y consistencia de los datos durante el intercambio de información con el servidor.		
	9	Se documentará la aplicación con diferentes manuales con el objetivo de garantizar el soporte de la misma.		
	10	Se desarrollara la aplicación utilizando el lenguaje de programación C Sharp con el entorno de desarrollo .NET.		
	11	La metodología de desarrollo a seguir es SXP y para la modelación se utilizará la herramienta Visual Paradigm3.4.		
	12	Para el correcto funcionamiento de la aplicación es necesario		

		tener instalado el Framework 2.0 de .NET o superior.		
	13	Los requisitos mínimos de hardware que se debe tener para el correcto funcionamiento de la aplicación son: 512 MB de RAM, microprocesador Pentium IV, tarjeta red Fast Ethernet		
	14	El código de la aplicación será liberado bajo la Licencia Pública General Reducida de GNU (LGPL).		
	15	Para el desarrollo de la aplicación se han establecido pautas para la codificación que permitan mantener un código uniforme y legible.		

## 2.5 Historias de Usuario y Tareas de Ingeniería

Las historias de usuario en la metodología de desarrollo SXP son las que describen las tareas que el sistema debe hacer, cuestión que depende en gran medida de las especificaciones realizadas por el cliente. Van a ser la guía para la construcción posterior de las pruebas funcionales comprobando de esta manera la correcta implementación de las historias de usuario.

A continuación se muestran en la (Tabla 3) la historia de usuario Establecer Conexión de la librería cliente C Sharp.

Tabla 3: Historia de Usuario Establecer conexión

Historia de Usuario	
Número: HU_1	Nombre Historia de Usuario: Establecer Conexión
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Rolando Betancourt Toucet	Iteración Asignada: 2

# Capítulo 2. Características, Análisis y Diseño del Sistema

<b>Prioridad en Negocio:</b> Muy alta	<b>Puntos Estimados:</b> 1
<b>Riesgo en Desarrollo:</b> Muy alta	<b>Puntos Reales:</b> 1
<b>Descripción:</b> La presente historia de usuario tiene como objetivo establecer la conexión con el servidor de jWebSocket. Para la cual se debe utilizar el esquema “ws” o “wss” en caso de utilizar cifrado ssl. Con el objetivo de establecer la comunicación con el servidor el cliente debe enviar una negociación inicial, que no es más que una actualización del protocolo http. El servidor debe analizar y en caso de cumplir con las especificaciones del protocolo responder al cliente correctamente en el tiempo previsto. Para mantener dicha conexión abierta se debe sostener un envío constante de paquetes entre el cliente y el servidor a intervalos predefinidos.	
<b>Observaciones:</b> Ninguna.	
<b>Prototipo de interfaz:</b> Ninguno.	

A continuación se describen las tareas de ingenierías asociadas a la historia de usuario Establecer Conexión. Estas tareas van a permitir una mayor organización y cumplimiento de las acciones que se llevan a cabo en dicha historia de usuario. En las tablas (Tabla 4, Tabla 5 y Tabla 6) se muestran los detalles de las tareas mencionadas.

Tabla 4: Tarea de Ingeniería Crear el Handshake

Tarea de Ingeniería	
<b>Número Tarea:</b> 1	<b>Número Historia de Usuario:</b> HU_1
<b>Nombre Tarea:</b> Crear el Handshake.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 2 días
<b>Fecha Inicio:</b> 12/09/11	<b>Fecha Fin:</b> 14/09/11
<b>Programador Responsable::</b> Rolando Betancourt Toucet	
<b>Descripción:</b> Implementar los métodos necesario para la creación del	

handshake de acuerdo a la versión del protocolo WebSocket mediante el cual se desea establecer la conexión con el servidor. Este handshake consiste en una cadena de texto que contiene los datos necesarios para realizar la actualización del encabezado del protocolo HTTP al protocolo WebSocket. Una vez establecida la conexión el servidor jWebSocket la interpreta como WebSocket y comienza a enviar información mediante el mismo.

Tabla 5: tarea de Ingeniería Procesar el Handshake

Tarea de Ingeniería	
<b>Número Tarea: 1.1</b>	<b>Número Historia de Usuario: HU_1</b>
<b>Nombre Tarea:</b> Procesar el Handshake	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 2 días
<b>Fecha Inicio:</b> 14/09/11	<b>Fecha Fin:</b> 16/09/11
<b>Programador Responsable::</b> Rolando Betancourt Toucet	
<p><b>Descripción:</b> Implementar los métodos necesario para procesar el handshake que es enviado por el servidor para responder la solicitud de conexión del cliente. Este handshake contiene una serie de valores dentro de los cuales se encuentra una clave en base64 que es enviada por el cliente, a la cual el servidor debe responder de forma satisfactoria siguiendo un algoritmo determinado. Si la información enviada por el servidor no coincide con los parámetros que fueron enviados por el cliente no se puede establecer la conexión.</p>	

Tabla 6: Tarea de Ingeniería Establecer la Conexión

Tarea de Ingeniería	
<b>Número Tarea: 1.2</b>	<b>Número Historia de Usuario: HU_1</b>
<b>Nombre Tarea:</b> Establecer la Conexión.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 3 días

# Capítulo 2. Características, Análisis y Diseño del Sistema

<b>Fecha Inicio:</b> 16/09/11	<b>Fecha Fin:</b> 19/09/11
<b>Programador Responsable::</b> Rolando Betancourt Toucet	
<b>Descripción:</b> Implementar los métodos necesario para establecer la conexión con el servidor. Para ello después de enviar el handshake hacia el servidor y procesar satisfactoriamente el handshake que envía el servidor se deben lanzar una serie de notificaciones y eventos para lograr su correcto funcionamiento. También se ejecuta una serie de funcionalidades para que después de establecida la conexión la aplicación sea capaz de escuchar todos los mensajes procedentes del servidor.	

A continuación se muestran en la (Tabla 7) la historia de usuario Enviar Token de la librería cliente C Sharp.

Tabla 7: Historia de Usuario Enviar Token

Historia de Usuario	
<b>Número:</b> HU_2	<b>Nombre Historia de Usuario:</b> Enviar token.
<b>Modificación de Historia de Usuario Número:</b> Ninguna.	
<b>Usuario:</b> Rolando Betancourt Toucet	<b>Iteración Asignada:</b> 3
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 6
<b>Riesgo en Desarrollo:</b> Alto	<b>Puntos Reales:</b> 6
<b>Descripción:</b> La presente historia de usuario tiene como objetivo enviar los tokens hacia el servidor jWebSocket. Dichos tokens pueden tener un formato binario o de texto, en dependencias del contexto de la aplicación. Si estos exceden el tamaño máximo permitido se deben fragmentar y enviarlos como tokens independientes.	
<b>Observaciones:</b> Ninguna.	
<b>Prototipo de interfaz:</b> Ninguno.	

A continuación se describen las tareas de ingenierías asociadas a la historia de usuario Enviar Token. Estas tareas van a permitir una mayor organización y cumplimiento de las acciones que se llevan a cabo en dicha historia de usuario. En las tablas (Tabla 8, Tabla 9, Tabla 10 y Tabla 11) se muestran los detalles de las tareas mencionadas.

Tabla 8: Tarea de Ingeniería Enviar Token Binario

Tarea de Ingeniería	
<b>Número Tarea: 2</b>	<b>Número Historia de Usuario: HU_2</b>
<b>Nombre Tarea:</b> Enviar Token Binario.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 2 semanas
<b>Fecha Inicio:</b> 20/09/11	<b>Fecha Fin:</b> 03/10/11
<b>Programador Responsable::</b> Rolando Betancourt Toucet	
<b>Descripción:</b> Implementar los métodos necesarios para enviar un token binario hacia el servidor jWebSocket. Esto consiste en hacer una abstracción a la información que se desea enviar, la cual se le conoce como Token. Para ello se debe crear un diccionario con los datos que posteriormente se convertirán en formato Json y finalmente en una secuencia de bytes para luego ser enviado mediante un canal TCP.	

Tabla 9: Tarea de Ingeniería Enviar Token de Texto

Tarea de Ingeniería	
<b>Número Tarea: 2.1</b>	<b>Número Historia de Usuario: HU_2</b>
<b>Nombre Tarea:</b> Enviar Token de Texto.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 2 semanas
<b>Fecha Inicio:</b> 04/10/11	<b>Fecha Fin:</b> 18/10/11

<b>Programador Responsable::</b> Rolando Betancourt Toucet
<b>Descripción:</b> Implementar los métodos necesarios para enviar un token de texto hacia el servidor jWebSocket. Esto consiste en hacer una abstracción a la información que se desea enviar, la cual se le conoce como Token. Para ello se debe crear un diccionario con los datos que posteriormente se convertirán en formato Json y finalmente en una secuencia Base64 para luego ser enviado mediante un canal TCP.

Tabla 10: Tarea de Ingeniería Enviar Token Fragmentado

Tarea de Ingeniería	
<b>Número Tarea: 2.2</b>	<b>Número Historia de Usuario: HU_2</b>
<b>Nombre Tarea:</b> Enviar Token Fragmentado.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 1 semana
<b>Fecha Inicio:</b> 19/10/11	<b>Fecha Fin:</b> 26/10/11
<b>Programador Responsable::</b> Rolando Betancourt Toucet	
<b>Descripción:</b> Implementar los métodos necesarios para enviar un token fragmentado hacia el servidor jWebSocket. Esto consiste en hacer una abstracción a la información que se desea enviar, la cual se dividirá en paquetes de tamaño definido por el usuario los que serán procesados en dependencia de su tipo, para luego ser enviados.	

Tabla 11: Tarea de Ingeniería Enviar Paquete

Tarea de Ingeniería	
<b>Número Tarea: 2.3</b>	<b>Número Historia de Usuario: HU_2</b>
<b>Nombre Tarea:</b> Enviar Paquete	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 1 semana

<b>Fecha Inicio:</b> 26/10/11	<b>Fecha Fin:</b> 31/10/11
<b>Programador Responsable::</b> Rolando Betancourt Toucet	
<p><b>Descripción:</b> Implementar los métodos necesarios para enviar un paquete hacia el servidor jWebSocket. Este paquete es la estructura básica que utiliza el servidor para comunicarse con el cliente. Todos los Tokens que se deseen enviar para el servidor se deben transformar en paquetes. Para transformarlos en paquete se debe realizar un complejo algoritmo siguiendo las especificaciones del protocolo WebSocket.</p>	

A continuación se muestran en la (Tabla 12) la historia de usuario Recibir Token de la librería cliente C Sharp.

Tabla 12: Historia de Usuario Recibir Token

Historia de Usuario	
<b>Número:</b> HU _3	<b>Nombre Historia de Usuario:</b> Recibir Token.
<b>Modificación de Historia de Usuario Número:</b> Ninguna.	
<b>Usuario:</b> Rolando Betancourt Toucet	<b>Iteración Asignada:</b> 4
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 4
<b>Riesgo en Desarrollo:</b> Alta	<b>Puntos Reales:</b> 4
<p><b>Descripción:</b> La presente historia de usuario tiene como objetivo recibir los tokens que son enviados desde un servidor jWebSocket. Debe ser capaz de darle el tratamiento adecuado para los diferentes tipos de token que pueden ser recibidos. En caso de ser un token tipo Ping se debe responder con un token tipo Pong. Si el token recibido es un fragmento, se debe realizar el proceso de ensamblado de dicho token.</p>	
<b>Observaciones:</b> Ninguna.	
<b>Prototipo de interfaz:</b> Ninguno.	



A continuación se describen las tareas de ingeniería asociadas a la historia de usuario Recibir Token. Estas tareas van a permitir una mayor organización y cumplimiento de las acciones que se llevan a cabo en dicha historia de usuario. En las tablas (Tabla 13 y Tabla 14) se muestran los detalles de las tareas mencionadas.

Tabla 13: Tarea de Ingeniería Recibir Token

Tarea de Ingeniería	
<b>Número Tarea: 3</b>	<b>Número Historia de Usuario: HU_3</b>
<b>Nombre Tarea:</b> Recibir Token.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 2 semanas
<b>Fecha Inicio:</b> 1/11/11	<b>Fecha Fin:</b> 15/11/11
<b>Programador Responsable::</b> Rolando Betancourt Toucet	
<b>Descripción:</b> Implementar los métodos necesarios para recibir un token desde el servidor jWebSocket. Consiste en recibir una secuencia de bytes con la cual se conformara la estructura básica de un token.	

Tabla 14: Tarea de Ingeniería Recibir e Interpretar Fragmentos de Token

Tarea de Ingeniería	
<b>Número Tarea: 3.1</b>	<b>Número Historia de Usuario: HU_3</b>
<b>Nombre Tarea:</b> Recibir e Interpretar Fragmentos de Token.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 2 semanas
<b>Fecha Inicio:</b> 16/11/11	<b>Fecha Fin:</b> 30/11/11
<b>Programador Responsable::</b> Rolando Betancourt Toucet	
<b>Descripción:</b> Implementar los métodos necesarios para recibir un token fragmentado desde el servidor jWebSocket. Consiste en recibir una serie de	

secuencia de bytes las cuales se ensamblaran para obtener como resultado la estructura básica de un token.

A continuación se muestran en la (Tabla 15) la historia de usuario Finalizar Conexión de la librería cliente C Sharp.

Tabla 15: Historia de Usuario Finalizar Conexión

Historia de Usuario	
<b>Número:</b> HU_4	<b>Nombre Historia de Usuario:</b> Finalizar Conexión.
<b>Modificación de Historia de Usuario Número:</b> Ninguna.	
<b>Usuario:</b> Rolando Betancourt Toucet	<b>Iteración Asignada:</b> 5
<b>Prioridad en Negocio:</b> Media.	<b>Puntos Estimados:</b> 1
<b>Riesgo en Desarrollo:</b> Medio.	<b>Puntos Reales:</b> 1
<b>Descripción:</b> La presente historia de usuario tiene como objetivo cerrar la conexión con el servidor jWebSocket. Para terminar dicha conexión el cliente debe enviar un token de cierre de acuerdo con la versión del protocolo seleccionado para terminar satisfactoriamente la conexión TCP con el servidor.	
<b>Observaciones:</b> Ninguna.	
<b>Prototipo de interfaz:</b> Ninguno.	

A continuación se describen las tareas de ingenierías asociadas a la historia de usuario Finalizar Conexión. Estas tareas van a permitir una mayor organización y cumplimiento de las acciones que se llevan a cabo en dicha historia de usuario. En la tabla (Tabla 16) se muestran los detalles de las tareas mencionadas.

Tabla 16: Tarea de Ingeniería Finalizar Conexión

## Tarea de Ingeniería

# Capítulo 2. Características, Análisis y Diseño del Sistema

<b>Número Tarea: 4</b>	<b>Número Historia de Usuario: HU_4</b>
<b>Nombre Tarea:</b> Finalizar conexión.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 1 semana
<b>Fecha Inicio:</b> 1/12/11	<b>Fecha Fin:</b> 8/12/11
<b>Programador Responsable::</b> Rolando Betancourt Toucet	
<b>Descripción:</b> Implementar los métodos necesarios para terminar la conexión con el servidor jWebSocket. Consiste en enviar un token de tipo cierre el cual será interpretado por el servidor y enviara la información necesaria para terminar la conexión en ambos lados sin ocasionar perdidas algunas.	

## 2.6 Plan de Releases

El plan de releases o iteraciones permite realizar las entregas intermedias y la entrega final del producto. Tiene como entrada la relación de Historias de Usuario definidas previamente. Para colocar una historia en cada iteración se tiene en cuenta la prioridad que definió el cliente para dicha historia. Como resultado de la priorización de historias de usuario se llegó a la siguiente planificación

A continuación en la (**Tabla 17**) se muestra la relación entre la iteración y la duración de las historias de Usuarios de la librería cliente C Sharp que se desean implementar en el proceso de desarrollo de este software.

Tabla 17: Plan de Releases

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
---------	-----------------------------	------------------------------	----------------

Iteración 2	En esta iteración se desarrollarán las funcionalidades para establecer la conexión con el servidor.	HU_1	1 semanas
Iteración 3	En esta iteración se desarrollarán las funcionalidades para enviar información hacia el servidor.	HU_2	6 semanas
Iteración 4	En esta iteración se desarrollarán las funcionalidades para enviar información hacia el servidor.	HU_3	4 semanas
Iteración 5	En esta iteración se desarrollarán las funcionalidades para cerrar la conexión con el servidor.	HU_4	1 semanas

## 2.7 Descripción de la Arquitectura

La arquitectura de software de la librería cliente C Sharp es una arquitectura basada en capas, se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada. La mayoría de las aplicaciones contiene código de presentación, código de procesamiento de datos y código de almacenamiento de datos. La arquitectura de las aplicaciones

difiere según como está distribuido este código y determina a gran escala como van a ser desarrollados los diferentes componentes del sistema.

Aunque la Librería cliente para desarrollar aplicaciones con el marco de trabajo `jQueryWebSocket` utilizando el lenguaje de programación `C Sharp` no presente código de presentación ni de almacenamiento de datos, se ha utilizado una arquitectura en capas donde los componentes de cada capa se comunican con los componentes de otra capa mediante un rol determinado utilizando para ello la interfaz de programación de aplicaciones (API) correspondiente a su responsabilidad.

Dentro de las ventajas que presentan la arquitectura en capas vale mencionar los beneficios por concepto de mantenimiento o actualización realizados a la aplicación en cuestión. Esto es posible porque de ocurrir algún cambio en el código de la librería, solo se vería afectada la capa en la que se realiza dicho cambio, sin necesidad de afectar el código en cualquier otra capa. También se pueden agregar nuevos componentes al sistema sin necesidad de modificar la API existente hasta el momento, basta con agregar otra capa y lograr una correcta separación de responsabilidades.

Para el desarrollo de la Librería Cliente `C Sharp` también fue definido el uso del protocolo de comunicación `WebSocket`. Esto viene dado por las ventajas que este protocolo brinda para el desarrollo de aplicaciones en tiempo real, las cuales fueron explicadas con anterioridad. Además teniendo en consideración que se desea lograr la integración entre las aplicaciones desarrolladas con el lenguaje de programación `C Sharp` y el marco de trabajo `jQueryWebSocket`.

## 2.8 Diseño con Metáforas

Como `SXP` está basada en `XP`, dicha metodología define un término llamado metáfora, lo cual según Martin Fowler es una historia compartida que describe como debería funcionar el sistema. El Diseño con metáforas es sencillamente el diseño de la solución más simple que pueda funcionar y ser implementado en un momento dado del proyecto; lo cual genera el artefacto conocido como Modelo de

Diseño, que a su vez está compuesto por un diagrama de paquetes.

En la (Fig. 2) se muestra el diagrama de paquetes propuesto para la librería cliente C Sharp:

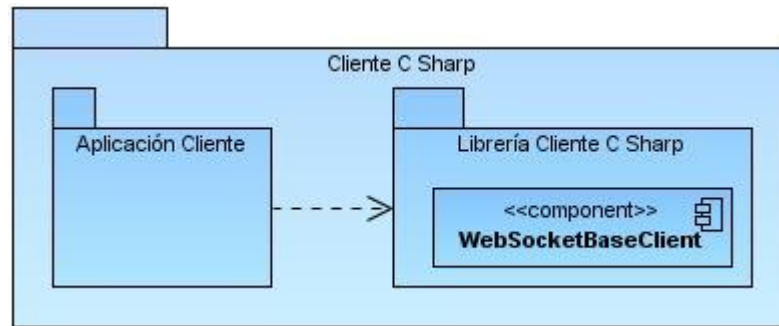


Fig. 2: Diagrama de Paquete de la librería cliente C Sharp

A continuación se muestra una información detallada de los paquetes y componentes que conforman el diagrama de paquete mostrado en la (Fig. 2)

- ✓ El paquete **Cliente C Sharp** está compuesto por dos paquetes que representan la estructura del directorio de la aplicación cliente.
- ✓ El paquete **Aplicación Cliente** representa la lógica de la aplicación, este paquete contiene todos los componentes relacionados con la aplicación cliente. El mismo hace referencia al paquete **Librería Cliente C Sharp** lo que le permite la utilización de la API de sus componentes para gestionar la conexión y comunicación con el marco de trabajo jWebSocket.
- ✓ El paquete **Librería Cliente C Sharp** es quien contiene todos los componentes que relacionados entre sí conforman una poderosa librería. Estos componentes brindan una serie de funcionalidades tales como la creación de Token, la gestión de la comunicación con el marco de jWebSocket y el envío y recibo de información entre ambos.

## 2.9 Diagrama de Componente

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y

ejecutable. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente, entre otros.

En la (Fig. 3) se muestra el diagrama de componentes propuesto para la librería cliente C Sharp donde se muestran sus principales componentes y la relación que existe entre ellos.

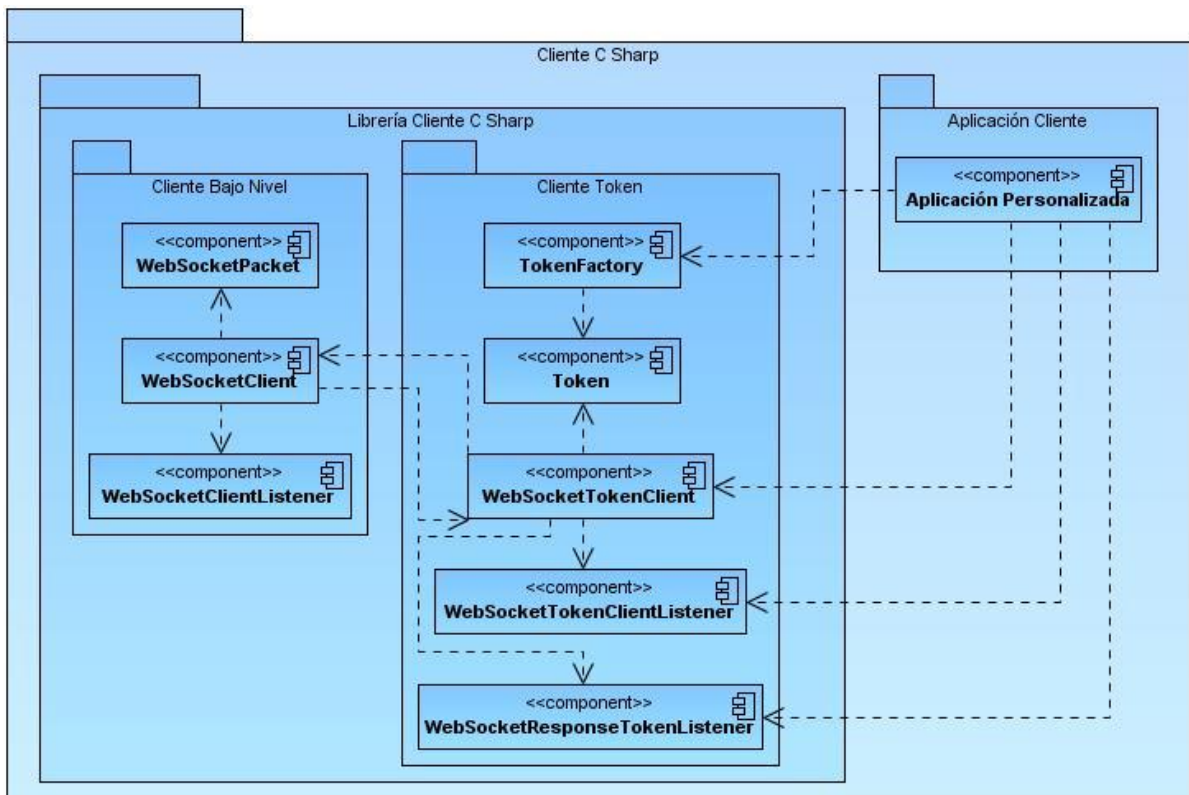


Fig. 3: Diagrama de Componentes de la librería cliente C Sharp

A continuación se brinda una descripción detallada de los principales componente de este diagrama para ayudar a tener una mejor comprensión y conocimiento del funcionamiento de la librería cliente C Sharp.

- ✓ El componente **Aplicación Personalizada** del paquete **Aplicación Cliente** no es más que la aplicación desarrollada en el lenguaje de programación C Sharp .Esta aplicación hace referencia al paquete **Librería Cliente C Sharp**

para poder utilizar todas las funcionalidades que este le brinda. Por lo cual podrá acceder a la API del componente **TokenFactory** para la creación de tokens, en los cuales almacenara toda la información que se desea enviara hacia el servidor. Con la API del componente **WebSocketBaseTokenCliente** podrá enviar los paquetes hacia el servidor y esperar la respuesta de los mismos.

- ✓ El componente **WebSocketBaseTokenClient** que pertenece al paquete **Librería Cliente C Sharp** es el encargado de gestionar toda la comunicación entre la aplicación cliente y el componente de bajo nivel **WebSocketbaseClient**. Este paquete hace función de intermediario procesando la información y transformándola en tokens para que llegue a la aplicación cliente con el formato correcto. Para esto brinda una API bien definida con una serie de funcionalidades que facilitara y agilizara el trabajo a la hora de desarrollar una aplicación.
- ✓ El componente **WebSocketBaseClient** que pertenece al paquete **Librería Cliente C Sharp** es el encargado de gestionar la conexión y comunicación con el marco de trabajo **jWebSocket**. Este paquete entre otras cosas es el encargado de transformar los tokens que vienen del cliente y transformarlos en paquetes para poder enviárselos al servidor. Cuando el servidor le responde este le transfiere la información al paquete **WebSocketBaseTokenClient** para que sea transformada en tokens y entregada a la aplicación cliente.
- ✓ El componente **TokenFactory** que pertenece al paquete **Librería Cliente C Sharp** es el encargado de la creación de un tipo de dato llamado **Token**. Este componente brinda una API bien definida con diversas funcionalidades para la creación de tokens de variadas formas y tipos. Este componente es utilizado por la aplicación cliente para poder encapsular los datos que desea enviar hacia el servidor.



## Conclusiones

En este capítulo se definieron las características y funcionalidades principales de la librería cliente para desarrollar aplicaciones con el marco de trabajo jWebSocket utilizando el lenguaje de programación C Sharp que se pretende desarrollar. Quedaron aprobados los requisitos funcionales y no funcionales necesarios para obtener una librería eficiente, segura y escalable. Se describieron las historias de usuario y tareas de ingeniería que se deben implementar, el plan de releases para establecer el cronograma de trabajo. También se realizó un análisis de la arquitectura a utilizar y un modelado del diagrama de paquetes y de componentes que permiten una mejor comprensión de la solución propuesta.

## Introducción

En el presente capítulo se describe la etapa de implementación y funcionamiento de la librería cliente C Sharp propuesta en el capítulo anterior, generándose el diagrama de despliegue de dicha solución. Se detallan los casos de pruebas o test funcionales realizados a las historias de usuario correspondientes a cada iteración, demostrando de esta forma el cumplimiento de los requerimientos definidos. Además, se exponen los resultados alcanzados hasta el momento y el aporte tanto social como económico de la solución.

### 3.1 Diagrama de Despliegue

El diagrama de despliegue es un artefacto que modela la arquitectura en tiempo de ejecución de un sistema. En él se indica la situación física de los componentes lógicos desarrollados, lo que significa que sitúa el software en el hardware que lo contiene. A continuación en la (Fig. 4) se muestra el diagrama de despliegue que representa la distribución física de la librería cliente C Sharp, donde este nodo representa un recurso de cómputo, siendo el mismo un procesador o dispositivo hardware que se necesita para el despliegue del sistema:



Fig. 4 Diagrama de Despliegue

A continuación se describe el Diagrama de Despliegue representado en la Fig. 4:

**Nodo Cliente:** Representa la aplicación cliente que consta de una solución personalizada que utiliza la librería cliente C Sharp para su funcionamiento.

## 3.2 Validación de la Investigación

La librería cliente C Sharp no es más que un conjunto de clases y funcionalidades que permite desarrollar aplicaciones en tiempo real en el lenguaje de programación C Sharp utilizando el marco de trabajo jWebSocket. La misma es de gran utilidad para los desarrolladores de este lenguaje debido al gran ahorro de tiempo que les permite a la hora de implementar cualquier aplicación que la utilice, así como la seguridad y confiabilidad de la misma.

Como la librería Cliente C Sharp es una solución tecnológica para realizar software, la mejor forma de probarla y validarla es desarrollando con ella aplicaciones complejas en dimensión y requerimientos. Sin embargo existen otros mecanismos enfocados a la validación por separado de los componentes que integran la solución que son las pruebas unitarias y pruebas funcionales.

Para la validación de la librería cliente C Sharp se llevó a cabo una estrategia de validación basada en 4 etapas fundamentalmente. En la primera se realizó una aplicación demostrativa con el objetivo de mostrar las funcionalidades de la librería cliente C Sharp para garantizar que realmente la misma funciona de manera correcta. En la segunda etapa se hacen los casos de pruebas funcionales alrededor de las historias de usuarios, que no son más que los requisitos funcionales que debe cumplir dicha solución.

Posteriormente en la tercera etapa se realiza un proceso de certificación de calidad del software por el grupo de calidad del centro de desarrollo de la facultad regional. El grupo de certificación de calidad del software hace la certificación principalmente basada en la funcionalidad, estandarización y limpieza del código. Se realiza la revisión de los artefactos documentales generados por la metodología SXP que fue seleccionada para el desarrollo de esta investigación.

Por ultimo en la cuarta etapa se decide poner a consideración mediante el método de experto la librería cliente C Sharp a Alexander Schulze líder de la comunidad de jWebSocket internacional y arquitecto principal del proyecto jWebSocket. En esta

etapa de validación se consigue que el líder del proyecto como experto internacional en la temática de la validación que tiene la aplicación realizada en la integración al marco de trabajo jWebSocket y la real usabilidad de esta para los usuarios de la comunidad.

Por último se realiza una valoración de la aplicación por parte del cliente Alexander Schulze. Esta etapa de validación tiene el objetivo de obtener una certificación de calidad de la librería cliente C Sharp desarrollada, comprobando su real usabilidad por parte de los usuarios de la comunidad jWebSocket. Seguidamente se describe con más detalle de cada una de las etapas de la estrategia de validación trazada a las cuales fue sometida dicha librería.

### 3.3 Aplicación demostrativa.

El demo C Sharp Client Demo permite probar las funcionalidades de la librería cliente C Sharp en una aplicación cliente. Esta aplicación posibilita crear una conexión a través del protocolo WebSocket con el marco de trabajo jWebSocket. Una vez conectada brinda la posibilidad de enviar mensajes, elegir las opciones de confiabilidad, iniciar sección, cerrar sección y desconectarse del servidor. A continuación se muestra un listado de sus funcionalidades y una imagen del demo en cuestión. Fig. 5

1. Campo donde se introduce la dirección URL del servidor al que desea conectarse.
2. Botón que permite abrir una nueva conexión (no abre sección nueva).
3. Botón que permite cerrar la conexión (no cierra sección).
4. Permite elegir las opciones de confiabilidad a la hora de establecer la conexión con el servidor.
5. Este campo permite especificar el tiempo de espera para reconectarse con el servidor.

6. Este campo permite especificar el tiempo de retraso para reconectarse con el servidor.
7. Campo donde se introduce el mensaje que se desea enviar.
8. Botón que permite el envío de mensajes hacia el servidor.
9. Botón que permite limpiar las trazas de las operaciones.
10. Botón que permite comenzar una conexión (abre sección nueva).
11. Botón que permite terminar una conexión (cierra sección).

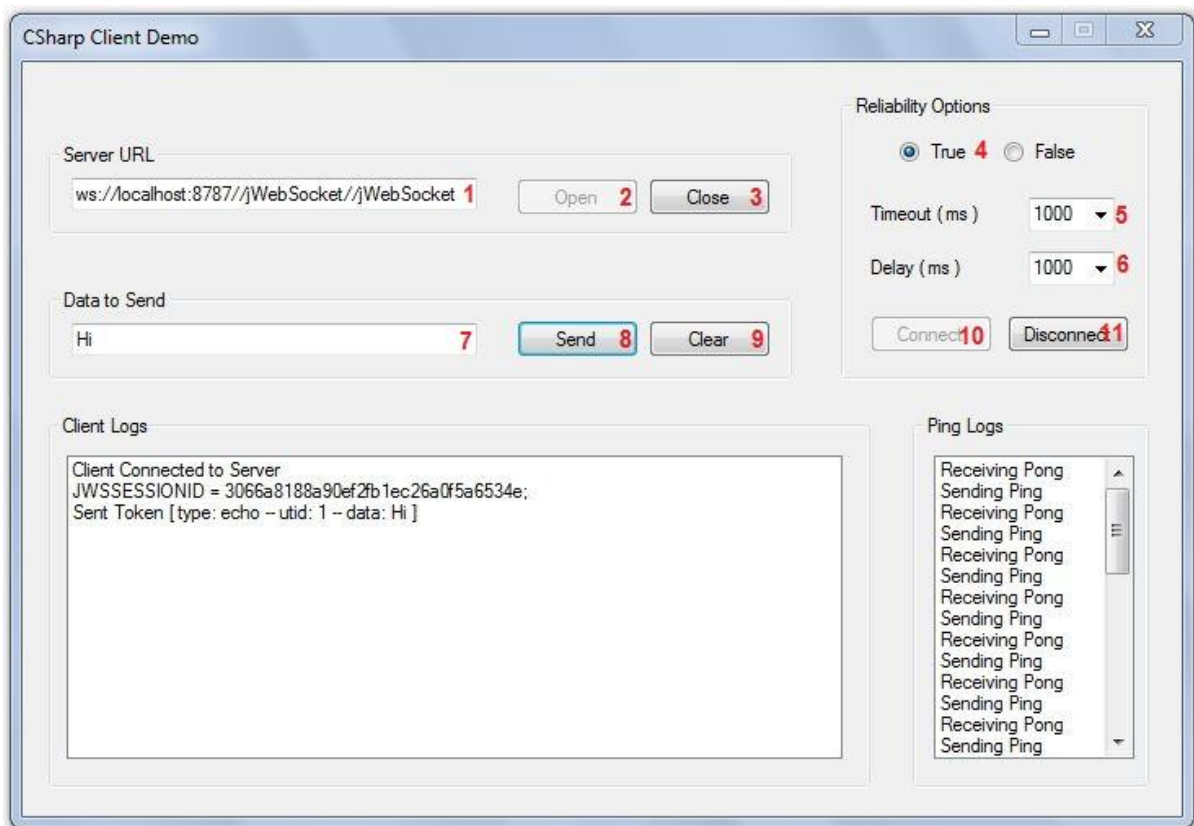


Fig. 5 Aplicación demostrativa

### 3.4 Casos de Prueba

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos especificados. Los resultados son observados y registrados y una evaluación es hecha de algún aspecto del sistema o componente.

Siendo un elemento que representa una revisión final de las especificaciones del diseño y de las codificaciones, de manera que encontrar un error que no había sido detectado con anterioridad es un objetivo fundamental.

Las pruebas funcionales son definidas por el cliente y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al cliente. La utilización de estas, proporcionan grandes ventajas, permitiendo a los programadores medir la calidad de su trabajo y garantizar la entrega de un producto con calidad y en correspondencia con las necesidades del cliente.

Se definieron casos de prueba para cada historias de usuario, En la (Tabla 18) se muestra la prueba funcional realizada a la historia de usuario Establecer conexión que consiste en lograr establecer la conexión con el servidor jWebSocket desde una aplicación cliente C Sharp.

Tabla 18 Caso de Prueba JWS-01-01

Caso de Prueba Funcional	
<b>Código Caso de Prueba:</b> JWS-01-01	<b>Nombre Historia de Usuario:</b> Establecer conexión.
<b>Nombre de la persona que realiza la prueba:</b> Rolando Betancourt Toucet	
<b>Descripción de la Prueba:</b> Esta prueba consiste en lograr establecer la conexión con el servidor jWebSocket desde una aplicación cliente escrita en C Sharp. Dicha conexión se pretende establecer sin utilizar las opciones de configuración de dicha conexión para que no se reconecte automáticamente con el servidor en caso de ocurrir algún tipo de interrupción por parte del servidor. Lo que significa que si ocurre algún fallo en el servidor la aplicación cliente lo notificara y continuara desconecta. Por otra parte si la aplicación cliente intenta conectarse al servidor y el mismo está fuera de funcionamiento, lo notificara y no intentara reconectarse en un futuro.	
<b>Condiciones de Ejecución:</b> Para ejecutar esta prueba debe estar corriendo un	

servidor `WebSocket`, al cual intentara conectarse la aplicación cliente. En el servidor debe estar configurado el *Plugin Echo*, con el cual la aplicación cliente intercambia información para poder realizar este test. La aplicación cliente se debe ejecutar de manera tal que no se activen las opciones de confiabilidad disponibles para la misma.

**Entrada / Pasos de ejecución:** Se ejecuta el servidor con sus configuraciones correspondientes. Se ejecuta la aplicación cliente. Se desmarcan las opciones de confiabilidad disponibles para la aplicación. Por último se presiona el botón (*Connect*) el cual es el encargado de crear la conexión con el servidor `WebSocket`.

**Resultado Esperado:** El cliente intenta conectarse con el servidor, Si el servidor se encuentra fuera de funcionamiento la aplicación cliente mostrara una notificación de error (*Could not establish the connection to the server*) y se mantendrá habilitado el botón (*Connect*) para poder reintentarlo en un futuro. Si la aplicación cliente se encuentra conectada con el servidor y se interrumpe la comunicación, dicha aplicación mostrar una notificación (*Client Close Connection*) y se habilitara el botón (*Connect*) para reintentar la conexión en un futuro.

**Evaluación de la Prueba:** Satisfactoria.

En la siguiente tabla (**Tabla 19**) se muestra la prueba funcional realizada a la historia de usuario Establecer conexión que consiste en establecer la conexión con el servidor `WebSocket` utilizando las opciones de confiabilidad.

Tabla 19 Caso de Prueba JWS-01-02

Caso de Prueba Funcional			
Código	Caso de Prueba:	Nombre Historia de Usuario:	Establecer

JWS-01-02	conexión.
<b>Nombre de la persona que realiza la prueba:</b> Rolando Betancourt Toucet	
<b>Descripción de la Prueba:</b> Esta prueba consiste en lograr establecer la conexión con el servidor <code>jWebSocket</code> desde una aplicación cliente escrita en C Sharp. Dicha conexión se pretende establecer utilizando las opciones de confiabilidad disponibles a la hora de establecer dicha conexión. Esto le brinda la posibilidad a la aplicación cliente de poder reconectarse automáticamente en caso de ocurrir cualquier fallo inesperado por parte del servidor. Si una aplicación cliente intenta conectarse a un servidor <code>jWebSocket</code> que no se encuentra corriendo en ese momento, dicha aplicación continuara tratando de establecer la conexión. Esta operación la realizara cada cierto intervalo de tiempo, el cual es definido junto con las opciones de confiabilidad. Si la aplicación cliente se encuentra conectado con el servidor y ocurre una interrupción en la comunicación la cual es ocasionada por el servidor, dicha aplicación tratara de reconectarse automáticamente una y otra vez hasta lograrlo. Para esto se configuran dos tiempos, el ( <i>delay</i> ) es el intervalo de tiempo que espera la aplicación cliente para reconectarse una vez interrumpida la conexión, y el ( <i>timeout</i> ) es el tiempo de espera de una respuesta por parte del servidor, si este tiempo es excedido la aplicación considera que se interrumpió la comunicación con el servidor y tratara de reconectarse una vez más.	
<b>Condiciones de Ejecución:</b> Para ejecutar esta prueba debe estar corriendo un servidor <code>jWebSocket</code> , al cual intentara conectarse la aplicación cliente. En el servidor debe estar configurado el <i>Plugin Echo</i> , con el cual la aplicación cliente intercambia información para poder realizar este test. La aplicación cliente se debe ejecutar de manera tal que se activen las opciones de confiabilidad disponibles para la misma, así como definir los tiempos correspondientes para dicha configuración.	
<b>Entrada / Pasos de ejecución:</b> Se ejecuta el servidor con sus configuraciones correspondientes. Se ejecuta la aplicación cliente. Se marcan las opciones de	



confiabilidad disponibles para la aplicación y se fijan los tiempos correspondientes para la misma. Por último se presiona el botón (*Connect*) el cual es el encargado de crear la conexión con el servidor `WebSocket`.

**Resultado Esperado:** El cliente intenta conectarse con el servidor, si el servidor se encuentra fuera de servicio por algún motivo, la aplicación cliente mostrara una notificación indicando que no se ha podido conectar (*Could not establish the connection to the server*) y otra notificando que está tratando de conectarse nuevamente (*Trying to Connect to the server*). De esta forma la aplicación cliente tratara de reconectarse a intervalos de tiempos definidos anteriormente por el usuario, hasta que se restaure el servidor y logre una conexión de forma satisfactoria. Si la aplicación cliente se encuentra conectada correctamente y ocurre algún suceso que provoca que se cierre la conexión por motivo del servidor, la aplicación mostrara las notificaciones pertinentes (*Could not establish the connection to the server*) (*Trying to Connect to the server*) y tratara de reconectarse nuevamente a intervalos de tiempo definidos por el usuario con anterioridad.

**Evaluación de la Prueba:** Satisfactoria.

En la siguiente tabla (**Tabla 20**) se muestra la prueba funcional realizada a la historia de usuario Enviar token que consiste en enviar un *token* hacia el servidor `WebSocket`

Tabla 20 Caso de Prueba JWS-02-01

Caso de Prueba Funcional	
<b>Código Caso de Prueba:</b> JWS-02-01	<b>Nombre Historia de Usuario:</b> Enviar token.
<b>Nombre de la persona que realiza la prueba:</b> Rolando Betancourt Toucet	

**Descripción de la Prueba:** Esta prueba consiste en enviar un *token* hacia el servidor *jWebSocket*. Para realizar esta tarea se debe seleccionar la información que se desea enviar. Dicha información es empaquetada siguiendo lo descrito en el protocolo *WebSocket* para luego encapsularla en una estructura de datos llamada *token*. El encapsulamiento varía de acuerdo a una serie de parámetros que se toman en cuenta y del tipo de información que se desea enviar para conformar un *token* binario o *token* de texto. Luego de procesada dicha estructura de datos se serializa y se convierte en formato *JSON* para luego en dependencia del tipo (texto o binario), generar un flujo de byte o una cadena en base64. Finalmente este flujo de información es enviado hacia el servidor a través de un canal TCP siguiendo todas las especificaciones y medidas de seguridad del protocolo *WebSocket*.

**Condiciones de Ejecución:** Para ejecutar esta prueba debe estar corriendo un servidor *jWebSocket* y la aplicación cliente conectada al mismo satisfactoriamente. También debe existir una correcta configuración del *Plugin Echo* para que pueda existir un intercambio de paquetes entre el cliente y el servidor. La aplicación cliente se debe ejecutar de manera tal que se activen o no las opciones de confiabilidad disponibles para la misma, así como definir los tiempos correspondientes para dicha configuración en caso de ser necesario.

**Entrada / Pasos de ejecución:** Se ejecuta el servidor con sus configuraciones correspondientes. Se ejecuta la aplicación cliente. Se marcan o no las opciones de confiabilidad disponibles para la aplicación y se fijan los tiempos correspondientes para la misma en caso de ser necesario. Por último se introduce la información que se desea enviar hacia el servidor y se presiona el botón (*Send*), el cual es el encargado de enviar dicha información hacia servidor *jWebSocket* para que sea procesada por el *Plugin Echo*.

**Resultado Esperado:** El cliente envía la información introducida con anterioridad hacia el servidor y lo notifica mostrando el *uuid* del paquete entre otros datos. En

caso que el cliente no se encuentre conectado satisfactoriamente no se habilitara la opción de enviar (*Send*).

**Evaluación de la Prueba:** Satisfactoria.

En la siguiente tabla (Tabla 21) se muestra la prueba funcional realizada a la historia de usuario Enviar token que consiste en enviar un *token* que exceda el tamaño máximo permitido hacia el servidor *jWebSocket*

Tabla 21 Caso de Prueba JWS-02-02

Caso de Prueba Funcional	
<b>Código Caso de Prueba:</b> JWS-02-02	<b>Nombre Historia de Usuario:</b> Enviar token.
<b>Nombre de la persona que realiza la prueba:</b> Rolando Betancourt Toucet	
<b>Descripción de la Prueba:</b> Esta prueba consiste en enviar un <i>token</i> que exceda el tamaño máximo permitido hacia el servidor <i>jWebSocket</i> . Para realizar esta tarea se debe seleccionar la información que se desea enviar. Dicha información es analizada y dividida en caso de necesitarlo para luego ser empaquetada siguiendo lo descrito en el protocolo <i>WebSocket</i> y finalmente encapsularla en una estructura de datos llamada <i>token</i> . El encapsulamiento varía de acuerdo a una serie de parámetros que se toman en cuenta y del tipo de información que se desea enviar para conformar un <i>token</i> binario o <i>token</i> de texto. Luego de procesada dicha estructura de datos se serializa y se convierte en formato <i>JSON</i> para luego en dependencia del tipo (texto o binario), generar un flujo de <i>bytes</i> o una cadena en <i>base64</i> . Este proceso se repite tantas veces como sea necesario, mientras la información que se desea enviar continúe siendo demasiado grande, la misma se dividirá en tamaños que se ajusten al máximo posible que se puede enviar. Finalmente estos flujos de información son enviados hacia el servidor como paquetes independientes a través de un canal <i>TCP</i> siguiendo todas las	

especificaciones y medidas de seguridad del protocolo WebSocket.

**Condiciones de Ejecución:** Para ejecutar esta prueba debe estar corriendo un servidor `jWebSocket` y la aplicación cliente conectada al mismo satisfactoriamente. También debe existir una correcta configuración del `Plugin Echo` para que pueda existir un intercambio de paquetes entre el cliente y el servidor. La aplicación cliente se debe ejecutar de manera tal que se activen o no las opciones de confiabilidad disponibles para la misma, así como definir los tiempos correspondientes para dicha configuración en caso de ser necesario.

**Entrada / Pasos de ejecución:** Se ejecuta el servidor con sus configuraciones correspondientes. Se ejecuta la aplicación cliente. Se marcan o no las opciones de confiabilidad disponibles para la aplicación y se fijan los tiempos correspondientes para la misma en caso de ser necesario. Por último se introduce la información que se desea enviar hacia el servidor y se presiona el botón (*Send*), el cual es el encargado de iniciar el proceso de fragmentación de la información antes de enviarla hacia el servidor `jWebSocket` para que sea procesada por el `Plugin Echo`.

**Resultado Esperado:** El cliente envía la información introducida con anterioridad hacia el servidor y lo notifica mostrando los `uuids` de los paquetes entre otros datos. En caso que el cliente no se encuentre conectado satisfactoriamente no se habilitara la opción de enviar (*Send*).

**Evaluación de la Prueba:** Satisfactoria.

En la siguiente tabla (**Tabla 22**) se muestra la prueba funcional realizada a la historia de usuario Recibir token que consiste en recibir un *token* que es enviado desde el servidor.

Tabla 22 Caso de Prueba JWS-03-01

## Caso de Prueba Funcional

# Capítulo3. Implementación y Validación del Sistema

<b>Código Caso de Prueba:</b> JWS-03-01	<b>Nombre Historia de Usuario:</b> Recibir token.
<b>Nombre de la persona que realiza la prueba:</b> Rolando Betancourt Toucet	
<b>Descripción de la Prueba:</b> Esta prueba consiste en recibir un <i>token</i> que es enviado desde el servidor. Dicho <i>token</i> puede ser enviado como respuesta al cliente o por cualquier otro motivo. Si consiste en una respuesta al cliente, sus valores de encabezado deben coincidir con los del <i>token</i> enviado inicialmente, o sea debe tener el mismo uuid y debe ser de tipo response. Cuando esta información llega al cliente es procesado por el mismo, o sea con sus datos de cabecera se confecciona un paquete que posteriormente transformara en una estructura de datos <i>token</i> . De esta forma el cliente puede brindar su información uniformemente. Si ocurre algún problema durante el procesamiento de dicha información se mostrara una notificación de error.	
<b>Condiciones de Ejecución:</b> Para ejecutar esta prueba debe estar corriendo un servidor <code>jWebSocket</code> y la aplicación cliente conectada al mismo satisfactoriamente. También debe existir una correcta configuración del <i>Plugin</i> Echo para que pueda existir un intercambio de paquetes entre el cliente y el servidor. La aplicación cliente se debe ejecutar de manera tal que se activen o no las opciones de confiabilidad disponibles para la misma, así como definir los tiempos correspondientes para dicha configuración en caso de ser necesario. Por último se debe enviar alguna información hacia el servidor para esperar su respuesta.	
<b>Entrada / Pasos de ejecución:</b> Se ejecuta el servidor con sus configuraciones correspondientes. Se ejecuta la aplicación cliente. Se marcan o no las opciones de confiabilidad disponibles para la aplicación y se fijan los tiempos correspondientes para la misma en caso de ser necesario. Por último se introduce la información que se desea enviar hacia el servidor y se presiona el botón ( <i>Send</i> ) para que la envíe y se quede esperando una respuesta.	

**Resultado Esperado:** El cliente recibe la respuesta desde el servidor satisfactoriamente mostrando la notificación del *token* que acaba de recibir, o sea el *uuid* y el tipo del *token* en cuestión.

**Evaluación de la Prueba:** Satisfactoria.

En la siguiente tabla (Tabla 23) se muestra la prueba funcional realizada a la historia de usuario Finalizar conexión que consiste en finalizar la conexión entre la aplicación cliente y el servidor *WebSocket*

Tabla 23 Caso de Prueba JWS-04-01

Caso de Prueba Funcional	
<b>Código Caso de Prueba:</b> JWS-04-01	<b>Nombre Historia de Usuario:</b> Finalizar conexión.
<b>Nombre de la persona que realiza la prueba:</b> Rolando Betancourt Toucet	
<b>Descripción de la Prueba:</b> Esta prueba consiste en finalizar la conexión entre la aplicación cliente y el servidor <i>WebSocket</i> . Para llevar a cabo esta prueba el cliente conforma un paquete con los datos necesarios para finalizar la conexión descritos en el protocolo <i>WebSocket</i> . Este paquete es procesado por el mismo y finalmente enviado hacia el servidor, el cual le dará un tratamiento diferente que al resto de los paquetes que recibe. En este caso el servidor lo procesa como un paquete de cierre y automáticamente comienza a cerrar todos los canales de comunicación con la aplicación cliente. Lo primero que hace es suspender el envío de paquetes <i>Pong</i> y posteriormente le envía otro paquete de cierre al cliente como respuesta a su solicitud de cierre. El cliente por su parte recibe dicho paquete y comprueba que el servidor ha aceptado su solicitud de cierre, por lo que también suspende el envío de paquetes <i>Ping</i> y cierra todos los canales de comunicación TCP con el servidor. Cuando ocurre la desconexión intencional por parte de la aplicación cliente el mismo no trata de reconectarse automáticamente	

con el servidor, puesto que la comunicación se ha suspendido por acuerdo de ambos lados.

**Condiciones de Ejecución:** Para ejecutar esta prueba debe estar corriendo un servidor `jWebSocket` y la aplicación cliente conectada al mismo satisfactoriamente. También debe existir una correcta configuración del *Plugin Echo* para que pueda existir un intercambio de paquetes entre el cliente y el servidor. La aplicación cliente se debe ejecutar de manera tal que se activen o no las opciones de confiabilidad disponibles para la misma, así como definir los tiempos correspondientes para dicha configuración en caso de ser necesario.

**Entrada / Pasos de ejecución:** Se ejecuta el servidor con sus configuraciones correspondientes. Se ejecuta la aplicación cliente. Se marcan o no las opciones de confiabilidad disponibles para la aplicación y se fijan los tiempos correspondientes para la misma en caso de ser necesario. Por último se presiona el botón (*Disconnect*) el cual comienza todo el proceso para terminar la conexión entre la aplicación cliente y el servidor.

**Resultado Esperado:** El cliente termina la conexión con el servidor y lo notifica de forma satisfactoria.

**Evaluación de la Prueba:** Satisfactoria.

### 3.5 Certificación de calidad.

Se puso a consideración del grupo de calidad del centro de desarrollo de la facultad regional el proyecto de software con los artefactos que fueron generados por SXP que son:

- ✓ Plantilla Modelo de Historia de Usuario del Negocio
- ✓ Plantilla lista de reserva del producto (LRP)
- ✓ Plantilla de Historia de usuario

- ✓ Plantilla de Arquitectura de Software SXP
- ✓ Plantilla Tarea de ingeniería
- ✓ Plantilla de Releases
- ✓ Plantilla Estándar de Código
- ✓ Plantilla Caso de prueba de Aceptación
- ✓ Plantilla Manual de Usuario
- ✓ Plantilla Manual de Instalación/Administración
- ✓ Plantilla Manual de Desarrollo

Este grupo de calidad llevo un proceso de revisión del código fuente donde se hizo énfasis en la estandarización de dicho código. Fue efectuado por los ingenieros Maidel Ojeda Castro y Domma Moreno Dargel en colaboración de Alexander Schulze quien también es asesor de tecnología, estos valoraron la calidad, limpieza y funcionamiento del código.

### **3.6 Valoración del Cliente.**

La presente investigación se puso a consideración del cliente Alexander Schulze. Toda la documentación generada así como el código implementado se subió al Subversion de jWebSocket Internacional, la documentación del manual de Desarrollador, del manual de Usuario y del manual de Instalación se realizó en inglés para que fluyera mejor la comunicación y así garantizar un mayor entendimiento de la aplicación por parte del cliente.

### **3.7 Resultados Obtenidos.**

La librería cliente C Sharp fue sometida a un conjunto de pruebas funcionales las cuales demuestran el cumplimiento de las funcionalidades que debe cumplir la misma. A través de la certificación de calidad realizada por terceros se comprobó que tanto la documentación como el código cuentan con la calidad, limpieza y estandarización que se necesita. Esto brinda la posibilidad que otros usuarios la



puedan usar y asegura además su disponibilidad ante futuras mejoras. El aval emitido por el grupo de calidad se encuentra en el Anexo 1.

El criterio emitido por el cliente permite garantizar que la librería pueda ser utilizada por la comunidad de jWebSocket. El certificado emitido por Alexander Schulze se encuentra en el Anexo 2. Al obtener resultados satisfactorios queda disponible la versión 1.0 de la librería cliente C Sharp, obteniéndose una Librería que cumple con todas especificaciones del cliente para la comunicación entre una aplicación cliente y el marco de trabajo jWebSocket.

### 3.8 Funcionalidades Obtenidas.

Entre las principales funcionales que posee librería cliente C Sharp en su versión 1.0 se encuentran:

- ✓ **Establecer la conexión con el marco de trabajo jWebSocket:** Esta funcionalidad permite realizar la conexión entre una aplicación desarrollada con el lenguaje de programación C Sharp y el marco de trabajo jWebSocket.
- ✓ **Enviar token de tipo texto hacia el marco de trabajo jWebSocket:** Esta funcionalidad permite enviar la información desde la aplicación cliente hacia el marco de trabajo jWebSocket utilizando una estructura de datos abstracta llamada *token*, y el tipo de datos que puede enviar es texto o sea una cadena en base64.
- ✓ **Enviar token de tipo binario hacia el marco de trabajo jWebSocket:** Esta funcionalidad permite enviar la información desde la aplicación cliente hacia el marco de trabajo jWebSocket utilizando una estructura de datos abstracta llamada *token*, y el tipo de datos que puede enviar es binario, o sea una cadena de bytes.
- ✓ **Enviar token fragmentado hacia el marco de trabajo jWebSocket:** Esta funcionalidad permite enviar información desde la aplicación cliente hacia el marco de trabajo jWebSocket utilizando la estructura de datos abstracta *token*. Cuando esta información es mayor que el tamaño máximo permitido

por el servidor entonces se envía dicho *token* en pequeños fragmentos que luego serán ensamblados en el servidor.

- ✓ **Recibir e Interpretar token entrantes desde el servidor de jWebSocket:** Esta funcionalidad permite recibir la información proveniente del servidor jWebSocket y procesarla para empaquetarla utilizando el tipo de dato abstracto *token*. En dicho formato es que se le brindara la información a la aplicación cliente que utiliza con esta librería.
- ✓ **Recibir e Interpretar fragmentos de token entrantes desde el servidor de jWebSocket:** Esta funcionalidad permite recibir la información proveniente del servidor jWebSocket y procesarla para empaquetarla utilizando el tipo de dato abstracto *token*. Cuando dicha información se encuentra fragmentada, entonces se emplea un potente algoritmo para ensamblarla y finalmente brindársela a la aplicación cliente que utiliza esta librería.
- ✓ **Cerrar la conexión con el marco de trabajo jWebSocket:** Esta funcionalidad es la encargada de enviarle un *token* con la información de cierre al marco de trabajo jWebSocket, el cual le responderá esa petición y suspenderá dicha conexión.

### 3.9 Aporte Social y Económico.

El desarrollo de una librería que garantice mejorar la productividad y confiabilidad de las aplicaciones clientes C Sharp para el marco de trabajo jWebSocket representa una herramienta indispensable para el desarrollo de aplicaciones clientes en tiempo real. La misma permite una completa integración de dichas aplicaciones con el marco de trabajo jWebSocket, así como un considerable ahorro de tiempo y recursos a las empresas y equipos de desarrollos que utilizan el lenguaje de programación C Sharp a la hora de desarrollar estas aplicaciones. Esto permite que los desarrolladores se centren en la lógica de la aplicación en vez de tratar de implementar el protocolo WebSocket para poder establecer la conexión con el servidor jWebSocket.

Todas estas ventajas que representan el poder utilizar esta librería en el desarrollo de aplicaciones clientes se traducen con un fuerte impacto positivo en la economía de las empresas que la utilizan. Todo esto relacionado a las mejoras, ahorro de tiempo y eficiencia que se pueden lograr en el equipo de desarrollo de este tipo de aplicaciones que utilizan tecnología de punta para la transmisión de datos en tiempo real.

En el ámbito social, la librería cliente para desarrollar aplicaciones con el marco de trabajo `jWebSocket` utilizando el lenguaje de programación C Sharp forma parte de un conjunto de soluciones del marco de trabajo `jWebSocket`, que se distribuye bajo la licencia de software libre LGPL. Esto permite que pueda ser utilizada sin costo alguno por los desarrolladores de C Sharp interesados en el desarrollo de aplicaciones en tiempo real. Esta librería ha sido desarrollada en la UCI y por tanto la misma se beneficia del soporte directo y capacitación a los estudiantes, profesores y trabajadores de esta institución. Lo cual representa un gran aporte a la comunidad de desarrolladores de dicha institución.

## Conclusiones

En este capítulo se realizó el diagrama de despliegue que indica la situación física de los componentes lógicos desarrollados. Además se realizaron casos de pruebas que guiaron la calidad de la aplicación desarrollada, se obtuvo el aval de certificación de calidad y el aval del experto Alexander Schulze. Demostrándose así que la librería cliente C Sharp se encuentra en óptimas condiciones para la comunicación entre una aplicación cliente y el servidor `jWebSocket`.

Con la realización de la presente investigación se dio respuesta a las preguntas científicas planteadas, obteniéndose de manera general las siguientes conclusiones:

- ✓ La fundamentación teórico-metodológica de la investigación permitió conceptualizar la comunicación mediante el protocolo WebSocket en aplicaciones clientes C#.
- ✓ Actualmente las librerías cliente permiten la conexión y transferencia de datos con varios servidores que utilizan tecnología WebSocket, pero no existe una librería cliente C Sharp para el marco de trabajo jWebSocket.
- ✓ La aplicación demostrativa C Sharp Client Demo se desarrolló utilizando la librería cliente C Sharp, permitiendo la conexión y transferencia de datos en tiempo real con el servidor jWebSocket. El uso de esta librería en las aplicaciones clientes garantiza un aumento de la productividad y confiabilidad en las mismas.
- ✓ Se comprobó la capacidad y potencialidad de la librería cliente C Sharp desarrolla a través de la estrategia de validación trazada para la presente investigación, garantizando que su código fuente y documentación poseen la calidad requerida, así como un alto nivel de usabilidad por parte de la comunidad de jWebSocket.

- ✓ Incluir en la próxima versión de la librería cliente C Sharp el mecanismo necesario para el control y administración de las cookies, mediante la cual se puede mantener una sesión para cada conexión con el servidor.
- ✓ Implementar el mecanismo de conexión segura SSL para brindarle más protección al flujo de información entre la aplicación cliente desarrollada con esta librería y el marco de trabajo jWebSocket.

**Apache Software Foundation. 2011.** Apache Subversion. *Apache Subversion*. [Online] 12 10, 2011. [Cited: 12 10, 2011.] <http://subversion.apache.org/>.

*Comet-HTML5-Websocket. Egli, P. R. (2011). 2011.* s.l. : Indigoo, 2011.

**Fette, I. and Melnikov, A. 2011.** The WebSocket Protocol. *Internet Engineering Task Force (IETF)*. [Online] 12 2011. [Cited: 02 17, 2012.] <http://tools.ietf.org/html/rfc6455>. 2070-1721.

**Figuroa, Roberth G, Solís, Camilo J and Cabrera, Armando A. 2011.** Entorno Virtual de Aprendizaje. [Online] 2011. [Cited: 12 13, 2011.] <http://eva.uci.cu/mod/resource/view.php?id=9303&subdir=/Metodologias/RUP>.

*Framework Approach for WebSockets. Schulze, A. (2011). 2011.* s.l. : Web Technologies & Internet Applications (WebTech 2011), 2011.

—. **Schulze, Alexander. 2011.** Web Technologies & Internet Applications (WebTech 2011) :

[http://dl.globalstf.org/index.php?page=shop.product\\_details&flypage=flypage\\_images.tpl&product\\_id=528&category\\_id=42&option=com\\_virtuemart&Itemid=4&vmcchk=1&Itemid=4](http://dl.globalstf.org/index.php?page=shop.product_details&flypage=flypage_images.tpl&product_id=528&category_id=42&option=com_virtuemart&Itemid=4&vmcchk=1&Itemid=4), 2011.

Grupo Soluciones Innova. [Online] [Cited: 12 11, 2011.]

<http://www.rational.com.ar/herramientas/roseenterprise.html>.

*Herramientas Case. Alfaro, Félix Murillo. 2011.* 2011, COLECCION CULTURA INFORMATICA.

*HTML5 Websockets and communication. Lubbers, Petter. 2010.* Java User Group Meeting : <http://www.slideshare.net/Kaazing/v2-peterlubberssfjugwebsocket>, 2010.

**Hybi. 2011.** The WebSocket protocol. *The WebSocket protocol*. [Online] septiembre 30, 2011. [Cited: diciembre 01, 2011.] <http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-17>.

**jwebsocket.org. 2011.** jwebsocket.org. [Online] 2011. <http://jwebsocket.org>.

**Kirkpatrick, Marshall. 2009.** Read Write Web. [Online] 9 22, 2009. [Cited: 02 21, 2012.] [http://www.readwriteweb.com/archives/explaining\\_the\\_real-time\\_web\\_in\\_100\\_words\\_or\\_less.php](http://www.readwriteweb.com/archives/explaining_the_real-time_web_in_100_words_or_less.php).

*Metodologías Tradicionales Vs. Metodologías Ágiles. Figuroa, Roberth G., Solís, Camilo J. and Cabrera, Armando A. 2011.* Loja : <http://www.docstoc.com/docs/102328746/articulo-metodologia-de-sw-formato>, 2011.

**Microsoft. 2011.** Microsoft. [Online] 2011.  
<http://www.microsoft.com/visualstudio/en-us>.

**Monodevelop. 2011.** Monodevelop. [Online] 2011. <http://monodevelop.com>.

**msdn. 2011.** msdn.microsoft.com. [Online] 2011. <http://msdn.microsoft.com/en-us/library/system.net.websockets.websocket%28v=vs.110%29.aspx>.

**NetBeans. 2011.** *NetBeans*. [Online] Oracle Corporation, 2011.  
<http://netbeans.org/features/index.html>.

**RapidSVN. 2011.** RapidSVN. [Online] 2011. [Cited: 12 13, 2011.]  
<http://www.rapidsvn.org/>.

*SXP, Metodología Ágil para el Desarrollo de Software. Peñalver, G, Meneses, A and García, S. 2010.* Chile : 1er congreso Iberoamericano de Ingeniería de Proyectos, 2010.

**Sybase. 2009.** *Client-Library Migration Guide*. 2009. DC36065-01-1550-01.

**team, TortoiseSVN. 2011.** TortoiseSVN. [Online] 2011. [Cited: 12 13, 2011.]  
<http://tortoisesvn.net/>.

**Visual Paradigm.** Visual Paradigm. [Online] [Cited: 12 11, 2011.] <http://www.visual-paradigm.com>.