Facultad Regional Mártires de Artemisa

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

Librería cliente para desarrollar aplicaciones con el marco de trabajo jWebSocket utilizando el lenguaje de programación Python.

Autor: Eylin Baydes González

Tutor: Ing. Leidy Laura Sánchez González

Artemisa, Cuba

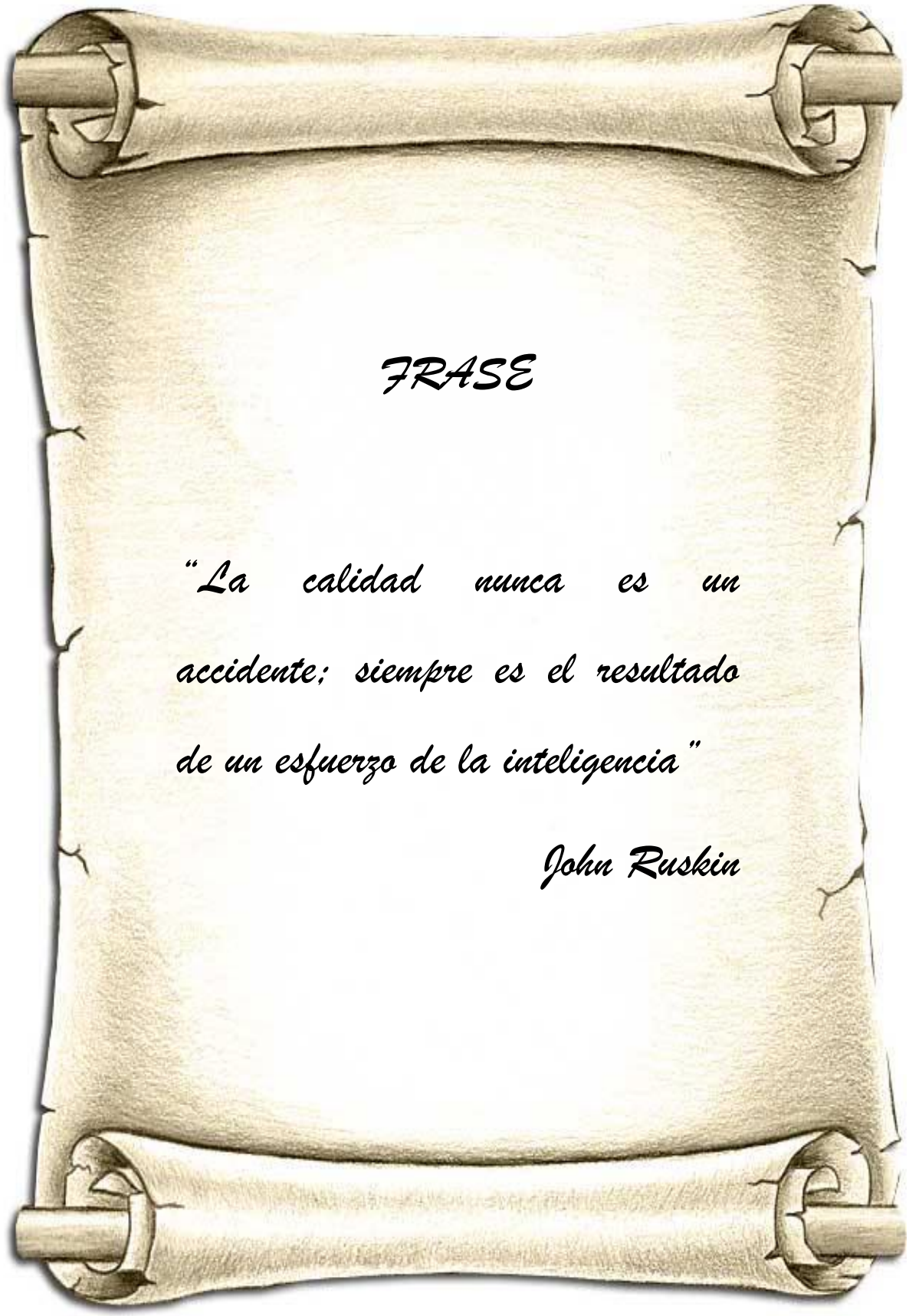
Junio 2012

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo a la Facultad Regional “Mártires de Artemisa” de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2012.

Eylin Baydes González	**Ing. Leidy Laura Sánchez**
_____	_____
Firma del Autor	Firma del Tutor



FRASE

"La calidad nunca es un accidente; siempre es el resultado de un esfuerzo de la inteligencia"

John Ruskin

AGRADECIMIENTOS

Me parece que en una hoja de este documento no me es suficiente para agradecerles a todas las personas que de una forma u otra han aportado un granito de arena para que yo cumpliera mis sueños de ser ingeniera. Pero bueno aquí les van mis agradecimientos:

Agradezco a mis padres, por estar siempre a mi lado a pesar de la distancia, por el apoyo incondicional, por el consejo oportuno, por el aliento y la esperanza que siempre me dan, incluso en los momentos más difíciles, por todo el amor, apoyo y confianza que me ha brindado siempre; a Silvia por haber sido una segunda madre para mí y estar a mi lado durante los últimos 4 años de mi carrera.

A mi novio por estar siempre a mi lado, incluso en los momentos más difíciles, por su apoyo incondicional y todo su cariño.

A mis amistades que siempre compartieron momentos tristes y alegres conmigo, que lloramos y reímos juntos, en especial, Geydis, Sosa, Dayani, Laury y Sady. Independientemente de las vueltas que de la vida, seguiremos siempre siendo amigas. A mi amigo TITO por aguantarme todas las veces que lo molestaba y lo mortificaba.

DEDICATORIA

Dedico este trabajo de diploma a mis padres por su incondicional e infinito amor, por toda la confianza que siempre depositaron en mí, por inspirarme en todo momento y guiarme a ser todo lo que soy.

RESUMEN

La Web se ha convertido en una medio de comunicación por excelencia. En ella se pueden encontrar diferentes servicios como son: herramientas de colaboración en línea, juegos en línea, redes sociales, audio y video streaming, entre otras. Estas aplicaciones requieren de una comunicación bidireccional y tiempo real.

El protocolo WebSocket propone un verdadero estándar de desarrollo para aplicaciones que requieren una comunicación bidireccional y en tiempo real en la Web. Aunque el protocolo solo fue diseñado para establecer la comunicación entre un navegador y un servidor, cualquier lenguaje de programación que soporte una comunicación por socket basada en TCP[1] puede implementar una librería cliente que posibilite la comunicación con el servidor.

El marco de trabajo jWebSocket es una tecnología que permite desarrollar aplicaciones con el protocolo WebSocket. La aplicaciones desarrolladas con jWebSocket brindad una arquitectura cliente–servidor. En la actualidad marco de trabajo jWebSocket solo cuanta con librerías cliente para el lenguaje de programación Java y JavaScript.

Hoy en día los desarrolladores del lenguaje de programación Python no pueden aprovechar las potencialidades del marco de trabajo jWebSocket porque este no brinda soporte para una librería cliente en este lenguaje. Crear esta librería es una tarea compleja porque requiere entrar en detalles del funcionamiento del protocolo WebSocket. Los métodos implementados pueden estar propensos a errores ya que no estarían respaldados, testeados ni documentados por una comunidad.

Con esta investigación se propone la creación de una librería cliente para desarrollar aplicaciones utilizando el lenguaje de programación Python. Esta librería facilitara en gran medida el desarrollo de dichas aplicaciones, así como debe permitir elevar los niveles de confiabilidad y productividad de las aplicaciones desarrolladas en Python y el marco de trabajo jWebSocket.

[1] TCP: Protocolo de Control de Transmisión.

ÍNDICE GENERAL

ÍNDICE DE TABLAS Y FIGURAS

INTRODUCCIÓN

La Web fue creada como un medio para compartir y consultar documentos en Internet. Hoy en día a superado con creses su objetivo inicial, evolucionando hasta convertirse en una gran red mundial. En esta residen millones de usuarios y se brindan diferentes servicios como son: audio, video streaming, juegos en línea, redes sociales y herramientas de desarrollo colaborativo por solo mencionar algunos. Este tipo de aplicaciones web requieren un alto grado de actualización constante de información en tiempo real, sin embargo el protocolo HTTP[2] es la base de todos los sistemas web en Internet. Este presenta un esquema de comunicación unidireccional y del tipo petición-respuesta, con una cabecera que puede alcanzar hasta 2Kb de información por mensajes. Esto genera congestiones en la red y le impide al servidor comunicarse con el cliente sin que este haya originado la petición. Solo la creación de nuevas tecnologías y grandes gastos en poderosos hardware han permitido establecer el tiempo real sobre la Web.

La tecnología WebSocket ha sido incorporada con el nuevo HTML5[3] y su primer borrador fue publicado en 2009, este representa uno de los mayores avances que desde su creación ha experimentado la Web. El protocolo brinda un verdadero canal bidireccional de comunicación que opera sobre una sola conexión por socket[4]. Propone un verdadero estándar para desarrollar aplicaciones web en tiempo real que puedan ser escalables sin altos costos de hardware. El hecho de contar con un socket nativo del navegador elimina los muchos problemas y complejidades que ocasionaba el uso de Comet para lograr el tiempo real y la comunicación bidireccional sobre la Web. Respecto a WebSocket el ingeniero de Google, Ian Hickson, líder de la especificación de HTML5 expresó: "*Reduciendo kilobytes de datos a 2 bytes... y reduciendo la latencia desde 150 ms hasta 50 ms es algo mucho más allá de ser marginal. De hecho, estos dos factores por si solos*

[2] HTTP: Protocolo de transferencia de hipertexto.

[3] HTML5: Lenguaje de Marcado de Hipertexto.

[4] Socket: Método para la comunicación entre un programa del cliente y un programa del servidor en una red.

son suficientes para hacer que WebSocket sea seriamente interesante para Google".[5]

La conexión que se establece entre el cliente y el servidor de WebSocket es una simple conexión basada en TCP. El protocolo WebSocket solo define las reglas sintácticas, semánticas y de sincronización en la comunicación. Aunque el protocolo fue diseñado para implementarse en los navegadores, cualquier aplicación cliente o servidora que soporte conexiones de tipo TCP pude implementar el protocolo y habilitarse para lograr la comunicación. Para establecer la conexión la parte cliente debe enviar el handshake[6] correspondiente a la especificación del protocolo, si el servidor entiende la versión del protocolo responde a la petición quedando establecida la conexión. Una vez estabilizada la conexión los datos pueden viajar libremente de forma bidireccional. En caso de que el servidor no entienda las especificaciones enviadas por el cliente en el handshake la conexión nunca es establecida.

En las aplicaciones que presentan un arquitectura de tipo cliente-servidor, al conjunto de clases o procedimientos que permiten o facilitan la comunicación con el servidor se les conoce comúnmente como librerías clientes (Client-Library). Resulta de suma importancia para las tecnologías usadas en el desarrollo de aplicaciones cliente-servidor contar con la mayor cantidad de librerías clientes posibles, ya que esto intensifica sus posibilidades de uso y existencia en el mercado. Las librerías-clientes brindan interfaces de programación bien definidas que rigen el comportamiento y los estándares para la comunicación con los servidores. Esto le permite al desarrollador abstraerse de los detalles inminentes y técnicos de la conexión y el intercambio de información a bajo nivel. Estas librerías clientes son usadas de forma nativas en cada uno de los lenguajes de programación.

El marco de trabajo jWebSocket es una tecnología de código abierto cuya licencia de distribución es LGPL[7]. Este proporciona los medios necesarios para desarrollar

[5] IETF website: http://www.ietf.org/mail-archive/web/hybi/current/msg00784.html

[6] Handshake: Es un término del idioma inglés que se refiere a la conversación protocolar que realizan los sistemas en arquitecturas cliente/servidor antes de establecer una conexión.

[7] LGPL :(Lesser General Publical License, antes Library General Publical License)

aplicaciones web robustas que necesiten de una comunicación rápida, bidireccional y en tiempo real, haciendo uso del protocolo WebSocket. Las aplicaciones desarrolladas con jWebSocket siguen la arquitectura cliente-servidor. La parte que concierne al servidor debe ser desarrollada usando el lenguaje de programación multiplataforma Java, mientras que la referente al cliente puede ser desarrollada en cualquier lenguaje que implemente sus librerías-clientes según la interfaz de comunicación definida por los desarrolladores de jWebSocket. En la comunicación con el servidor, actualmente se soportan los formatos JSON, XML, CVS y cada librería-cliente puede implementar todos o solo los formatos mediante los cuales desee intercambiar información con el servidor.

Python es un lenguaje de programación multiparadigma. Esto significa que no obliga a adoptar un estilo único de programación en particular, permitiendo así programación orientada a objetos, programación imperativa y programación funcional. Presenta una licencia PSFL[8] completamente libre y compatible con GPL[9] que permite el uso gratuito del lenguaje aún en la realización de aplicaciones empresariales. La sintaxis y semántica del lenguaje se enfocan en lograr la facilidad, lectura y buena estructuración del código. Estas características hacen de Python un lenguaje altamente productivo y eficiente para la construcción de aplicaciones. Es un lenguaje interpretado con soporte para la mayoría de los sistemas operativos que sean comerciales o no. En los últimos años se ha vuelto tan popular que se encuentra hoy preinstalado en la mayor parte de las distribuciones de la familia GNU Linux.

Teniendo en cuenta los elementos anteriores, se describe la siguiente **situación problemática:**

Hoy en día los desarrolladores del lenguaje de programación Python no pueden aprovechar las potencialidades del marco de trabajo jWebSocket porque este no brinda soporte para una librería cliente en este lenguaje. Esto obliga a los desarrolladores de este lenguaje que deseen desarrollar aplicaciones con el servidor de jWebSocket a invertir tiempo y recursos en implementar dicha librería

[8] PSFL: Python Software Foundation License.

[9] GPL: General Public Licence.

alejándose así de la lógica de la aplicación. Crear esta librería es una tarea compleja porque requiere entrar en detalles del funcionamiento del protocolo WebSocket. Los métodos implementados pueden estar propensos a errores ya que no estarían respaldados, testeados ni documentados por una comunidad. Esto atenta contra la productividad y confiabilidad de las aplicaciones que se desarrollen en Python y el marco de trabajo jWebSocket.

De la problemática existente surge la siguiente interrogante que define el **problema de investigación:** ¿Cómo mejorar la productividad y confiabilidad de las aplicaciones en tiempo real con el lenguaje de programación Python utilizando el marco de trabajo jWebSocket?

Para orientar la investigación se cuenta con el **objeto de estudio:** La comunicación mediante el protocolo WebSocket para aplicaciones clientes en Python, delimitándose como **campo de acción:** La comunicación mediante el protocolo WebSocket en aplicaciones clientes en Python para el marco de trabajo jWebSocket.

Para realizar la investigación se trazó el **objetivo general:** Desarrollar una librería que permita incrementar los niveles de productividad y confiabilidad de las aplicaciones clientes en Python 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 para aplicaciones clientes en Python?
- ✓ ¿Cuál es la situación actual de la comunicación mediante el protocolo WebSocket para aplicaciones clientes en Python?
- ✓ ¿Cómo desarrollar una librería que permita incrementar los niveles de productividad y confiabilidad de las aplicaciones clientes en Python para el marco de trabajo jWebSocket?
- ✓ ¿Qué resultados se obtienen tras aplicar las estrategias de validación a la librería cliente desarrollada para incrementar la productividad y confiabilidad de las aplicaciones clientes en Python para el marco de trabajo jWebSocket?

Para guiar de forma organizada el presente trabajo se definieron las siguientes **tareas de la investigación**:

1) Fundamentación teórico-metodológica de la comunicación mediante el protocolo WebSocket en aplicaciones clientes en Python.
2) Análisis de la situación actual de la comunicación mediante el protocolo WebSocket en aplicaciones en Python.
3) Desarrollo de una librería que permita incrementar la productividad y confiabilidad de las aplicaciones clientes en Python para el marco de trabajo jWebSocket.
4) Comprobar la capacidad de la librería desarrollada para incrementar la productividad y confiabilidad de las aplicaciones clientes en Python para el marco de trabajo jWebSocket.

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

Variable Independiente: Librería para aplicaciones clientes en Python para el marco de trabajo jWebSocket.

Variables Dependientes:

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

Para darle solución a las tareas asignadas se utilizaron los siguientes **métodos de científicos**:

Métodos teóricos:

- ✓ **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 este análisis, las experiencias y las sugerencias que pudieran ser incorporadas a la tesis.

- ✓ **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:** Se utiliza para analizar la trayectoria completa del proceso de las librerías clientes en Python, 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 compresión del problema y desarrollar un modelo para la aplicación a desarrollar a partir de la situación problemática.

Con este trabajo de diploma se pretende alcanzar los **Posibles Resultados:**

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

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

Capítulo 1. Fundamentación Teórica: Se realiza la fundamentación teórico-metodológica de la investigación. Se expone un estudio del estado del arte sobre las librerías clientes existentes en la actualidad y las principales herramientas y tecnologías utilizadas.

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.

Capítulo 3. Implementación y Validación del Sistema: 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.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción.

En este capítulo se elabora la base teórico-metodológica 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 un estudio de las metodologías, tecnologías y herramientas a tener en cuenta para el desarrollo de la solución propuesta.

1.1 Marco Teórico.

La Web ha evolucionado a lo largo de los años, acumulando experiencias y novedosos avances tecnológicos. En sus inicios solamente se hablaba de documentos y sitios web puramente estáticos con imágenes y texto, con el paso del tiempo los contenidos pasaron a ser administrados por usuarios y la información era dinámica y social. En la actualidad ya se habla de una Web que representa el triunfo del mundo virtual y la inteligencia de las máquinas, accesible desde cualquier dispositivo, llegando a convertirse en un medio de comunicación omnipresente y en el que grandes flujos de información provenientes de las más diversas aplicaciones y servicios sociales de Internet, son procesados en tiempo real.

Tiempo real en informática se define como el conjunto de hardware y software que están sujetos a estrictas restricciones de tiempo desde la ocurrencia de un evento hasta la respuesta del sistema al mismo. Generalmente los tiempos de respuesta en tiempo real son entendidos en el orden de los milisegundos y algunas veces en microsegundos. Asociado al tiempo real se ha definido los elementos de un sistema informático en tiempo real. Entre las principales definiciones se encuentra la dada por Ben-Ari que plantea: “Un sistema informático en tiempo real debe soportar las restricciones de tiempo asociadas a las respuestas del sistema aunque el mismo se encuentre altamente sobrecargado” (Ben-Ari, 1990)

El tiempo real en la Web se rige teniendo en cuenta las bases teóricas establecidas por la definición anterior, aunque incorpora elementos propios de la Web. Es por

ello que se plantea que tiempo real en la Web es un paradigma en el que los usuarios reciben la información tan pronto como esta se encuentra disponible, en lugar de tener que consultar periódicamente la fuente de información por actualizaciones. (Kirkpatrick, 2009)

WebSocket es la tecnología estandarizada por la W3C[10] y la IEETF[11] para desarrollar aplicaciones web en tiempo real. Este protocolo permite conexiones bidireccionales entre un cliente y un servidor. Consiste en un mecanismo de handshake 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 y full-duplex con el servidor en vez de tener que realizar múltiples conexiones HTTP. (Hybi, 2011)

Las conexiones bidireccionales en la Web permiten que no solo sea el cliente quien inicia una solicitud al servidor sino que también el servidor puede hacer solicitudes al cliente sin una petición previa del mismo. Esta característica beneficia a las aplicaciones con inmensos ahorros en cuestiones de consumo de ancho de banda en la red y carga en los servidores de aplicaciones. Por su parte la característica full-duplex se refiere a un canal de comunicación o dispositivo que es capaz de hacer transmisiones de datos simultáneamente en dos direcciones. (The Linux Information Project, 2005)

La contante evolución de la Web ha posibilitado el vertiginoso avance de las tecnologías que dan soporte al desarrollo y despliegue de las aplicaciones. En este aspecto, juegan un papel primordial los servidores de aplicación y marcos de trabajo.

jWebSocket es un servidor de aplicaciones y a su vez un marco de trabajo creado específicamente para soportar el desarrollo de aplicaciones web en tiempo real utilizando el protocolo WebSocket. (Schulze, 2011) Esta tecnología posee librerías clientes en los lenguajes Java y JavaScript. Contar con un mayor número de librerías clientes agrega posibilidades tecnológicas al marco de trabajo a la hora de desarrollar aplicaciones.

[10] W3C: World Wide Web Consortium.

[11] IETF: Internet Engineering Task Force.

Se define librería cliente como un grupo de interfaces de programación utilizadas para desarrollar aplicaciones clientes. Las librerías clientes proporcionan bloques genéricos para la construcción de aplicaciones clientes distribuidas. (Sybase, 2009) En el caso de jWebSocket las librerías clientes garantizan que desde múltiples plataformas varios clientes puedan contribuir o realizar acciones sobre una misma aplicación.

1.2 Tendencias Actuales.

El 23 de abril del año 2009 fueron publicadas las especificaciones del protocolo WebSocket. Desde entonces nuevas tecnologías y otras ya existentes comenzaron a dar soporte con el tiempo a este nuevo protocolo. Servidores de renombre y con años de experiencias como Grizzly[12], Jetty[13] y Netty[14] actualmente ya dan soporte al nuevo protocolo. WebSocket proponen un modo de comunicación diferente sobre la Web y esto ha propiciado el surgimiento de nuevas tecnologías centralizadas en el uso del protocolo para desarrollar aplicaciones. Estas nuevas tecnologías por lo general pretenden ser una completa plataforma, por lo cual brindan soluciones para las dos partes (cliente/servidor) que intervienen en el intercambio de información dentro de una aplicación web.

Kaazing WebSocket Gateway es una de las nuevas tecnologías arraizadas al surgimiento del protocolo WebSocket. Kaazing es una pasarela que habilita la comunicación bidireccional entre un cliente y un servidor basado en TCP, esto hace posible que varias tecnologías puedan ser usadas en el lado del servidor como son JMS[15], JMX[16], IMAP[17], incluso el servidor de Jabber. En el lado del cliente también poseen librerías de distintas tecnologías como JavaScript, Flex, Microsoft Silverlight y Java/JavaFX. Esta tecnología ofrece una gran flexibilidad a la hora de desarrollar pero tiene como gran inconveniente su licencia, que es netamente comercial.

---

[12] http://grizzly.java.net

[13] http://jetty.codehaus.org/jetty

[14] http://www.jboss.org/netty

[15] JMS: Jail Management System.

[16] JMX: Java Management Extensions.

[17] IMAP: Protocolo de Acceso a Correo.

jWebSocket es otra de las nuevas tecnología de código abierto con licencia LGPL, que pretende ser más que una pasarela y convertirse en un completo marco de trabajo para el desarrollo de aplicaciones en tiempo real usando el protocolo WebSocket. Esta tecnología trae incorporado su propio servidor y aunque es extensible mediante motores, resulta obligatorio su uso en la implementación de una aplicación. Actualmente jWebSocket contiene motores para TCP, Nio, Netty, Tomcat, Grizzly y Jetty. Mientras que para desarrollar las aplicaciones del lado del cliente se cuentan con librerías para JavaScript, Java y Android.

No todas las nuevas tecnologías se han enfocado en crear una solución completa para desarrollar software utilizando WebSocket. Existen proyectos que se han encargado de crear librerías clientes para diversos lenguajes que cumplan con las especificaciones básicas del protocolo. Una vez que estas librerías brindan la implementación del protocolo pueden ser utilizadas para comunicarse con el servidor en el lenguaje de programación pertinente. Ejemplos de estas librerías son Weberknecht[18] que es una librería para java, Socket.IO[19] para desarrolladores en JavaScript, SocketRocket[20] entre otras. En el caso específico de Python se han desarrollados varias librerías para WebSocket como son el caso de ws4py[21] y WebSocket-client[22].

Cuando se inicia la investigación de esta tesis solo se encuentra disponible para Python la librería WebSocket-client. Esta librería está desactualizada dado que solo soporta el borrador hixie76[23], el proyecto era solo el código fuente subido al hithub[24] sin página oficial. Incluso cuando esta librería no estaba de acorde a la última versión del protocolo y presentaba algunos problemas de implementación se reutilizaron algunos fragmentos de código en la solución propuesta de la presente investigación.

[18] http://code.google.com/p/weberknecht/

[19] http://socket.io/

[20] http://corner.squareup.com/2012/02/socketrocket-websockets

[21] www.defuze.org/archives/271-ws4py-websocket-client

[22] https://github.com/liris/websocket-client

[23] http://tools.ietf.org/html/draft-hixie-thewebsocketprotocol-76

[24] https://github.com/

1.3 Metodologías a utilizar en el desarrollo de la aplicación.

Para el desarrollo de la solución propuesta se hace necesaria la selección de una metodología de desarrollo acorde a las características propias del proyecto y que a su vez garantice la eficiencia y calidad del proceso de desarrollo de software. Con este objetivo fue analizada las metodologías ágiles SCRUM, XP y SXP. De cada una de estas se exponen sus características principales, lo cual permite adquirir los elementos necesarios para determinar cuál es la más adecuada para guiar el proceso de desarrollo de software de la solución propuesta.

1.3.1 Metodologías Ágiles:

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. Entre los principales métodos ágiles se encuentra XP y Scrum.

Programación Extrema (XP):

XP (Extreme Programming) es un enfoque de la ingeniería del software formulado por Kent Beck. Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (INTECO, 2009)

Ventajas:

- ✓ Apropiado para entornos volátiles.
- ✓ Estar preparados para el cambio, significa reducir su coste.
- ✓ Planificación más transparente para nuestros clientes, conocen las fechas de entrega de funcionalidades. Vital para su negocio.

- ✓ Permite tener realimentación de los usuarios muy útil.
- ✓ La presión está a lo largo de todo el proyecto y no en una entrega final.

Desventajas:

- ✓ Delimitar el alcance del proyecto con nuestro cliente. (Figueroa, et al., 2010)

SCRUM:

Scrum es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software. Es orientado a las personas más que a los procesos. Emplea una estructura de desarrollo ágil: incremental basada en iteraciones y revisiones. Está diseñado especialmente para adaptarse a los cambios en los requerimientos, por ejemplo en un mercado de alta competitividad.

Scrum tiene un conjunto de reglas muy pequeño y muy simple y está basado en los principios de inspección continua, adaptación, auto-gestión e innovación.

Scrum controla de forma empírica y adaptable la evolución del proyecto, empleando las siguientes prácticas de la gestión ágil: Revisión de las Iteraciones, Desarrollo incremental, Desarrollo evolutivo, Auto-organización, Colaboración. (Figueroa, et al., 2010)

Una de las mayores ventajas de Scrum es que es muy fácil de entender y requiere poco esfuerzo para comenzar a usarse (INTECO, 2009)

El proceso de desarrollo Scrum se compone de 5 fases: (omercade, 2010)

- ✓ Planes de lanzamientos.
- ✓ Distribución, revisión y ajuste de los estándares de producto.
- ✓ Sprint.
- ✓ Revisión del Sprint.
- ✓ Cierre.

SXP:

SXP es una metodología compuesta por las metodologías SCRUM y XP que ofrece una estrategia tecnológica. Se parte de la introducción de procedimientos ágiles que permiten actualizar los procesos de software para el mejoramiento de la

actividad productiva. Fomenta el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo y ayudando al líder del proyecto a tener un mejor control del mismo.

SCRUM es una forma de gestionar un equipo de manera que trabaje de forma eficiente y de tener siempre medidos los progresos, de forma que se sepa por donde se anda. XP más bien es una metodología encaminada para el desarrollo; consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto.

SXP 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 la fase donde se realiza la implementación y construcción del sistema hasta que esté listo para ser entregado, luego de un conjunto de iteraciones de desarrollo.
- ✓ **Entrega:** Fase donde se pone en marcha el producto desarrollado y se genera la documentación necesaria para hacer la entrega al cliente.
- ✓ **Mantenimiento:** En esta se realiza el soporte para los problemas que pueda presentar el software durante su despliegue.

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 añada una nueva funcionalidad.

SXP está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido 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. Ayuda a que trabajen todos juntos en la misma dirección, con un objetivo claro, permitiendo además seguir de forma clara el avance de las tareas a realizar, de

forma que los jefes pueden ver día a día como progresa el trabajo. (Peñalver, et al., 2010)

Teniendo en cuenta las características de la metodología SXP y dadas las particularidades de la presente investigación, la estrecha comunicación que se mantiene con el cliente y el hecho de ser considerado un proyecto de corta duración, donde toda la responsabilidad del proceso de diseño e implementación es asumida por el autor de este investigación. Se selecciona SXP como la metodología para el desarrollo de la solución propuesta.

1.4 Herramienta y Tecnologías a utilizadas en el desarrollo de la solución.

1.4.1 Lenguajes Modelado.

Lenguaje de Modelación Unificado (UML):

Hoy en día, UML está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante UML es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código. El UML es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson y está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Los diagramas más comunes son: diagramas de clases, objetos, de casos de uso, estados, secuencias, actividades, colaboraciones y componentes; además los mismos permiten examinar un sistema desde distintos puntos de vista.

Un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema. Es importante recalcar que en un modelo UML no es necesario que aparezcan todos los diagramas. De hecho, la mayoría de los modelos UML contienen un subconjunto de los diagramas que se han mencionado.

Objetivos de UML:

- ✓ Es un lenguaje de modelado de propósito general que pueden usar todos los modeladores.

- ✓ No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática.
- ✓ Ser tan simple como fuera posible pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesita construir.

Cabe señalar que UML no se utiliza para lograr el cumplimiento del proyecto, pero si mejora la organización y rapidez del desarrollo del software ya que permite una cohesión entre los procesos y herramientas. Al igual que es importante dejar claro que este lenguaje permite especificar procesos y métodos pero no describir los mismos, pues solo se trata de una notación. Se utilizó este lenguaje porque permite al analista generar diseños que capturen sus ideas de una forma convencional y fácil de comprender para comunicarlas a otras personas, además por ser un lenguaje para visualizar, especificar, construir y documentar los artefactos del sistema que se pretende desarrollar. Se escogió también porque permite realizar una verificación del modelo realizado, además de expresar mediante una forma gráfica y fácil de interpretar, las funcionalidades y las descripciones de los procesos de negocio. Se seleccionó también porque posibilita especificar cuáles son las características de un sistema antes de su construcción. (Schmuller, 2010)

Se selecciona UML porque es el lenguaje utilizado por la herramienta de modelado Visual Paradigm seleccionada para la presente investigación y permite al mismo tiempo modelar todos los artefactos definidos por la metodología SXP.

1.4.2 Herramienta CASE:

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Los estados en el Ciclo de Vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación. (Menéndez, 2011)

Rational Rose:

Permite completar los flujos fundamentales de RUP[25] y se ha convertido en una de las mejores opciones por la notación estándar que brinda, para especificar, visualizar y construir productos software y sistemas, por lo que está en la avanzada en cuanto al desarrollo de UML. Rational Rose es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros del equipo a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. También Rose permite que los diseñadores puedan modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto. Pero aún con todos estos beneficios presenta la desventaja de una licencia con un alto costo de adquisición. (Menéndez, 2011)

Visual Paradigm:

Visual Paradigm para UML es una herramienta profesional que facilita el modelado del ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML soporta las últimas versiones del mismo y la Notación y Modelado de Procesos de Negocios. Ayuda a una más rápida construcción de aplicaciones de calidad y a un menor coste ya que permite generar código desde diagramas y generar documentación.

Visual Paradigm ofrece distintas funcionalidades como:

- ✓ Entorno de creación de diagramas para UML 2.0.
- ✓ Diseño centrado en casos de uso y enfocado al negocio generando un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa en su versión profesional e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de múltiples versiones para cada necesidad.
- ✓ Disponibilidad de integrarse en los principales IDE.

[25] RUP: Proceso Unificado de Desarrollo.

- ✓ Disponibilidad en múltiples plataformas (Windows, Linux, etc.) (Visual Paradigm International Ltd, 2011)

Se decidió trabajar con el Visual Paradigm 6.4 for UML por su robustez, usabilidad y portabilidad. (Visual Paradigm International Ltd, 2011) Además por ser una herramienta que se conoce bastante en la universidad, se cuenta con la licencia para su utilización y soporta el ciclo completo del proceso de desarrollo de software propuesto por la metodología SXP. Utiliza UML como lenguaje de modelado, que tiene como propósito visualizar, especificar, construir y documentar proyectos de software. Además ejecutarse sobre diferentes sistemas operativos lo que le confiere la característica de ser multiplataforma.

1.4.2 Lenguajes usados para el desarrollo de la solución

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90, cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Se trata de un lenguaje de programación multiparadigma, pues soporta orientación a objetos, programación imperativa, programación orientada a aspectos y en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico, es fuertemente tipado y multiplataforma.

El entorno de ejecución detecta muchos de los errores de programación que escapan al control de los compiladores y proporciona información muy rica para detectarlos y corregirlos. Puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objeto. Posee un rico juego de estructuras de datos que se pueden manipular de modo sencillo. Una ventaja fundamental es la gratuidad de su intérprete, el cual tiene versiones para prácticamente cualquier plataforma en uso: sistemas PC bajo Linux, sistemas PC bajo Microsoft Windows, sistemas Macintosh de Apple, etc. (González Duque, 2010)

Ventajas del uso de Python**:**

- ✓ **Propósito general:** Se pueden crear todo tipo de programas.
- ✓ **Multiplataforma:** Hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.

- ✓ **Interpretado:** No se debe compilar el código antes de su ejecución. En realidad sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador.
- ✓ **Interactivo:** Dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudarnos a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.
- ✓ **Orientado a Objetos:** La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables. Además, Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.
- ✓ **Funciones y librerías:** Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que se pueden importar en los programas para tratar temas específicos.
- ✓ **Sintaxis clara:** Tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.
- ✓ **Mixto:** Se puede integrar de manera "fácil" con otros lenguajes de programación.
- ✓ **Gratuito:** Una ventaja fundamental de Python es la gratuidad de su intérprete, se puede descargar desde la página web: http://www. Python.org. (Dayley, 2007)

El uso del lenguaje de programación Python en esta investigación está regido principalmente por la petición realizada por parte del cliente. El cliente identifico las potencialidades de este lenguaje y expreso la necesidad de contar con una librería cliente para permitir el desarrollo de aplicaciones clientes en este lenguaje.

1.4.3 Entorno Integrado de Desarrollo (IDE):

NetBeans:

Herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas de una forma más fácil. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación como Ruby, C/C++ o PHP. Con NetBeans 7.0.1 es posible además, utilizar la librería jQuery de JavaScript. Otro aspecto importante es que Netbeans es multiplataforma, permitiendo así, que funcione en diversos sistemas operativos como Windows, Mac, Linux o Solaris.

Con NetBeans es posible elaborar potentes aplicaciones de escritorio, para la Web y para dispositivos portátiles (móviles o Pocket PC). La programación mediante este se realiza a través de componentes de software modulares, también llamados módulos que le aportan gran funcionalidad y versatilidad. (Módulo para la realización de presentaciónes web reusable sobre Moodle, 2011)

Geany:

Geany es un Entorno de Desarrollo Integrado (IDE), pequeño y rápido con base a características y pocas dependencias con otros paquetes o entornos de escritorio. (VARGAS, 2011) Es compatible con Linux, Mac OS X, BSD (todas sus variantes) y Windows. En todos los casos con una interfaz agradable, dividida en 3 secciones: panel lateral con el árbol de carpetas y documentos abiertos, sección principal para el código y panel inferior para los mensajes de la aplicación, compilación, entre otros.

Geany ofrece soporte para crear y editar en varios de los lenguajes más utilizados, como son C, C++, Java, Python, Pascal, SQL o HTML, con herramientas interesantes como código auto expandible (ideal para achicar y expandir funciones) y auto completado de etiquetas HTML.

Algunas características:

- ✓ Resaltado de sintaxis.
- ✓ Finalización de código.
- ✓ Plegado de Código (separado por funciones, etc.).

- ✓ Construir la terminación / fragmentos.
- ✓ Cierre automático de etiquetas XML y HTML.
- ✓ Llamada a consejos/trucos.
- ✓ Soporte para múltiples lenguajes como C, Java, PHP, HTML, Python, Perl, Pascal o FreeBasic.
- ✓ Listas de símbolos y el nombre del símbolo de auto-realización.
- ✓ Código de navegación.
- ✓ Gestión de proyectos sencillos.
- ✓ Interfaz para Plugins. (Klew, 2011)

Las características anteriormente expuestas sobre NetBeans y Geany demuestran las potencialidades de ambos para el desarrollo de aplicaciones en Python. Sin embargo, para el desarrollo de la solución propuesta se selecciona como entorno integrado de desarrollo a Geany debido a que es IDE sumamente rápido y ligero. Teniendo en cuanta su última versión 0.20 trae incorporado disimiles plugin y con características suficientes para llevar a cabo el desarrollo de la solución.

1.4.4 Herramienta Control de versiones:

Los sistemas de control de versiones son herramientas software que permiten la automatización de tareas comunes sobre archivos (guardar, recuperar, registrar, borrar, identificar, mezclar) manteniendo una correcta gestión sobre las versiones de la información almacenada.

1.4.4.1 Servidor de control de versiones:

Git:

Git es un sistema de control de versiones distribuido, es completamente libre y de código abierto. No depende de acceso a la red o un repositorio central. Enfocado a la velocidad, uso práctico y manejo de proyectos grandes. Creado por Linus Torvalds, el creador del núcleo Linux. Entre sus características se encuentra eficiencia, desarrollo no lineal, limpieza del espacio de trabajo, autenticación criptográfica del repositorio, desaparición del número de versión del repositorio y a diferencia de otros sistemas de control de versiones Git almacena el fichero completo en lugar de diferencias respecto al anterior. (Paez , 2011)

Subversión:

Subversión es un sistema de control de versiones libre y de código fuente abierto. Es decir, Subversión maneja ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos.

Subversión puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. (Collins-Sussman, y otros, 2004)

Características importantes de Subversión:

- ✓ 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.

Considerando las características de Subversion, la experiencia que se posee con esta herramienta por parte del autor de este trabajo y las políticas de software libre establecidas por la Universidad de las Ciencias Informáticas, se decide su utilización para garantizar una mayor seguridad y disponibilidad de los datos en la presente investigación.

1.4.4.2 Cliente de control de versiones:

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

RapidSVN:

Una de las herramientas más populares para el control de versiones bajo el Sistema Operativo Linux es RapidSVN, el cual es un cliente gráfico para Subversión. Permite el acceso a direcciones SVN, subir y descargar contenido y sincronizarlo con el servidor original, comprobar su estado, crear y fusionar direcciones, y otras funcionalidades, además de disponer de un manual en línea bastante completo.

Es una herramienta sencilla y eficiente, que proporciona una interfaz fácil de acuerdo a las características de Subversión. Su eficiencia radica en su sencillez de uso para los principiantes, pero lo suficientemente flexible como para aumentar la productividad para los usuarios experimentados de Subversión. RapidSVN se caracteriza además por ser portable, pues funciona en cualquier plataforma en la que Subversión y wxWidgets (plataforma especializada en el desarrollo de aplicaciones multiplataforma en lenguaje C++, que soporta varios sistemas operativos, es libre y distribuido bajo licencia LGPL modificada.) se ejecuten: Linux, Windows, Mac OS 9 / X, Solaris, etc. La última versión de ésta herramienta es RapidSVN 1.5.5. (Pérez Bisset, y otros, 2009)

TortoiseSVN

TortoiseSVN es un cliente para Subversión, implementado de forma muy práctica, que se integra perfectamente al entorno de Windows. Es un software libre, distribuido según la licencia GNU/GPL; fácil de usar y que brinda un acceso más rápido y visual a la generalidad de las funciones de Subversión. Este cliente es ejecutable sobre Windows 2000 SP2, Windows XP o superiores. La última versión es TortoiseSVN 1.4.7, que vio la luz el 5 de enero de 2008.

Este cliente proporciona fácil acceso a los comandos de Subversión, pues añade su propio menú en el menú contextual del explorador donde están disponibles los comandos de este sistema de control de versiones.

TortoiseSVN proporciona un mecanismo flexible para integrar a cualquier web de seguimiento de fallos de sistema. Es estable y posee una herramienta de informes personalizados de choque, lo que ayuda a corregir los errores mucho más rápido. Es una herramienta rápida y descriptiva, cuya documentación está disponible en varios idiomas y formatos; además de poseer una nutrida guía de preguntas

frecuentes y los problemas con todas las respuestas disponibles. (Pérez Bisset, y otros, 2009)

Se analizaron algunas de las herramientas, como por ejemplo TortoiseSVN y RapidSVN. A pesar de todas las ventajas y funcionalidades que ofrece TortoiseSVN, este no se integra bien con sistemas GNU/Linux donde se lleva a cabo la implementación de la librería cliente para python. Por este motivo se selecciona RapidSVN que es una herramienta libre y fácil de usar. Este permite acceder a direcciones SVN, subir y descargar contenido, sincronizarlo con el servidor original, comprobar su estado, crear y fusionar direcciones.

Conclusiones del capítulo.

Una vez finalizado el presente capítulo se han dejado evidenciadas las bases teóricas que sustentarán el proceso de desarrollo de la solución del problema planteado. El estudio de librerías clientes para WebSocket y su implementación en el lenguaje de programación Python. Se estudiaron y eligieron las principales herramientas y metodología de desarrollo para iniciar la implementación de la librería cliente para el marco de trabajo de jWebSocket.

CAPÍTULO 2: CARÁCTERÍSTICAS, ANÁLISIS Y DISEÑO DEL SISTEMA.

Introducción.

En el presente capítulo se detalla cómo ha sido concebido el sistema y como debe funcionar, se hace énfasis en sus características distintivas así como la distribución y planificación del proyecto a nivel de roles. Se especifican brevemente los artefactos construidos que forman parte de la metodología de desarrollo SXP propuesta en el capítulo anterior. Se realiza el modelado del negocio con el objetivo de esclarecer el contexto de la solución. Se describen los requisitos funcionales y no funcionales del sistema, las historias de usuario y las tareas de ingeniería asociadas a las mismas.

2.1 Propuesta de Solución.

Como propuesta de solución es presentada una librería cliente para desarrollar aplicaciones utilizando el lenguaje de programación Python y el marco de trabajo jWebSocket del lado del servidor. Esta librería incorpora un valor agregado al marco de trabajo jWebSocket y aumenta considerablemente sus posibilidades de uso teniendo en cuenta la gran comunidad con la que cuenta el lenguaje de programación Python.

Para la implementación de la librería no se utiliza ningún marco de trabajo o tecnología de terceros. Todas las funcionalidades implementadas están basadas en propiedades nativas del lenguaje de programación Python. Esto permite incrementar la portabilidad de la misma sin mayor contratiempo, así como la reutilización de la librería en proyectos mayores sin problemas de licencias. Dentro de estas propiedades del lenguaje se destacan por su amplio uso las librerías socket, thread, y JSON. Es de vital importancia que la versión de Python que se tome para desarrollar de soporte a estas librerías.

La solución permite la creación de aplicaciones que interactúen con el servidor de jWebSocket. Se implementaron las interfaces propuestas por el marco de trabajo jWebSocket y la abstracción del modelo de datos que propone este. La transferencia no ocurre mediante simples mensajes si no que se utiliza una estructura común denominada token. Sin embargo para gestionar la transferencia a bajo nivel con el servidor se implementó el protocolo WebSocket sobre el socket

TCP nativo de Python. Por tanto este componente puede ser utilizado para lograr la conexión y el intercambio de mensajes con cualquier otro servidor que soporte WebSocket de forma nativa.

Esta librería proporciona un gran número de ventajas para los desarrolladores de Python que deseen desarrollar aplicaciones en tiempo real utilizando el marco de trabajo jWebSocket o interactuar con aplicaciones ya existentes. Los desarrolladores pueden ya contar con una librería que brinda una interfaz de programación bien definida, documentada y testeada por la comunidad, permitiéndoles centrarse en la lógica de las aplicaciones clientes en cuestión. Se destacan en este aspecto la facilidad que posee el servidor de jWebSocket en hacer extensible su uso mediante motores, permitiendo a las aplicaciones web y de escritorio compartir información en tiempo real, teniendo solamente al servidor de jWebSocket como intermediario.

2.2 Planificación del Proyecto por Roles.

SXP define diferentes roles para lograr un exitoso resultado en el proceso de desarrollo de software. A continuación 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	Dirige y controla las tareas del equipo. Toma las decisiones finales. Participa en la selección de objetivos y requerimientos. Controla el progreso y da seguimiento a cada iteración. Evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes.	Alexander Schulze
Cliente	Participa en las tareas que involucran la lista de reserva del producto.	Alexander Schulze
Analista	Escribe las historias de usuario y las pruebas funcionales para validar su implementación.	Eylin Baydes González
Arquitecto	Se vincula directamente con el analista y el diseñador debido a que su trabajo tiene que ver con la estructura y el diseño en grande del sistema. Ayuda en el diseño de las metáforas.	Eylin Baydes González
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.	Eylin Baydes González
Encargado de Prueba	Es el encargado de ayudar al cliente a escribir las pruebas	Eylin Baydes González

	funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.	
Diseñador	Encargados del diseño del sistema; así como el de los prototipos de interfaces, máximos responsables de la realización del diseño de las metáforas y supervisan el proceso de construcción.	Eylin Baydes González

2.3 Modelo de Historia de Usuario del Negocio.

La creación del Modelo de Historias de Usuario del Negocio es uno de los artefactos más importantes embebidos en la metodología SXP. En este se realiza la especiación 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. El diagrama de historias de usuario del negocio que se muestra a continuación se corresponde con los objetivos de la presente investigación. En este se describen los requerimientos principales asociados al funcionamiento, que en cuanto a conectividad se refiere presentan los sistemas informáticos que interactúan con servidores de WebSocket. Este rasgo distintivo del negocio que está presente de forma circunstancial en todas las aplicaciones que utilizan la tecnología WebSocket es la base principal que ha suscitado el desarrollo de esta investigación.

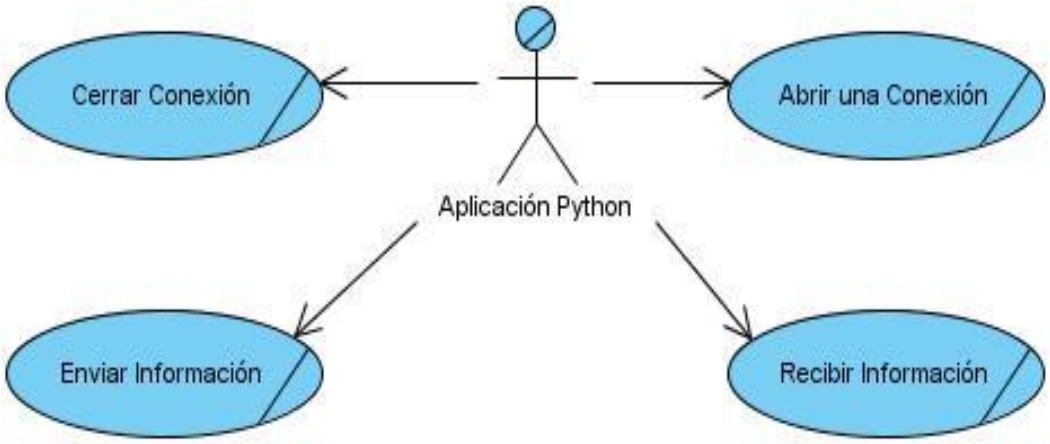


Figura 1: Modelo de Historia de Usuario del Negocio.

Aplicación Python: No es muy común encontrar un sistema informático como único actor del negocio. Pero en este caso resulta que es una aplicación desarrollada en Python la única beneficiara de la implementación de una librería cliente de WebSocket para este lenguaje, donde es la librería el sistema informático que se desea desarrollar. A modo de resumen es una aplicación el actor del negocio al cual están dirigidas todas las historias de usuarios.

2.4 Lista de Reserva del Producto (LRP).

La lista de reserva del producto (LRP) es una lista priorizada donde se especifican los requisitos funcionales y no funcionales para tener en cuenta en el futuro proyecto a desarrollar, o sea cuenta con los requisitos que debe llevar la librería cliente establecidos por el cliente, y se define en la etapa de planificación del proyecto. A continuación se muestra la lista priorizada contenida en el LRP perteneciente a la solución propuesta.

Tabla 2: Lista de Reserva del Producto.

Prioridad	Ítem *	Descripción	Estimación	Estimado por
Muy Alta				
	1	Establecer la conexión.	1 Semana	Analista
Alta				
	2	Enviar token binario.	2 semanas	Analista
	3	Enviar token de texto.	2 semanas	Analista

	4	Enviar token fragmentado con un tamaño determinado.	2 semanas	Analista
	5	Recibir e Interpretar token entrantes.	2 semanas	Analista
	6	Recibir e Interpretar fragmentos de token.	2 semanas	Analista
Media				
	7	Finalizar la conexión.	1 Semana	Analista
RNF (Requisitos No Funcionales)				
	8	Se desarrollara la aplicación utilizando el lenguaje de programación Python empleando el Framewok jWebSocket y el IDE Geany 0.20.		
	9	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.		
	10	El lenguaje de programación que debe ser utilizado para la implementación es Python.		
	11	La metodología de desarrollo a seguir es SXP y para la modelación se utilizará la herramienta Visual Paradigm 3.4.		
	12	El código de la aplicación será liberado bajo la Licencia Pública General Reducida de GNU (LGPL).		

	13	Garantizar la integridad y consistencia de los datos durante el intercambio de información con el servidor.		
	14	Para el desarrollo de la aplicación se han establecido pautas para la codificación que permitan mantener un código uniforme y legible.		
	15	Se documentará la aplicación con diferentes manuales con el objetivo de garantizar el soporte de la misma.		

2.5 Historias de Usuario y Tareas de Ingeniería.

Las historias de usuarios en la metodología de desarrollo SXP son las descriptoras de las tareas que el sistema debe hacer, cuestión que depende en gran medida de las especificaciones realizadas por el cliente. Se escriben con un lenguaje natural y con palabras concisas para no exceder su tamaño en unas pocas líneas de texto. Van a ser la guía para la construcción posterior de las pruebas de aceptación comprobando de esta manera la correcta implementación de las historias de usuario.

A continuación se dan a conocer las distintas historias de usuarios que están presentes en la librería cliente y las tareas asociadas a cada historia de usuario.

Tabla 3: Descripción de la HU_1 establecer conexión con el servidor.

Historia de Usuario	
Número: HU_1	**Nombre Historia de Usuario:** Establecer conexión con el servidor.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Eylin Baydes González	**Iteración Asignada:** 2
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 1

Riesgo en Desarrollo: Alta	**Puntos Reales:** 1
Descripción: La presente historia de usuario tiene como objetivo establecer la conexión con el servidor de WebSocket. Para establecer la conexión el host del servidor se especifica usando el esquema de url para WebSocket ws://host o wss://host para ssl. El cliente debe enviar el handshake inicial y verificar que la respuesta del servidor este acorde con la versión del protocolo especificado. Si el servidor responde dentro del tiempo establecido la conexión debe permanecer abierta.	
Observaciones:	
Prototipo de interface: Ninguna.	

Tabla 4: Descripción de la Tarea de Ingeniería 1.1 correspondiente a la HU_1.

Tarea de Ingeniería	
Número Tarea: 1.1	**Número Historia de Usuario:** HU_1
Nombre Tarea: Estudiar la Arquitectura General de Clientes jWebSocket.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 día
Fecha Inicio: 31/10/11	**Fecha Fin:** 31/10/11
Programador Responsable: Eylin Baydes González	
Descripción: La presente tarea tiene como objetivo diseñar un cliente de alto nivel que cumpla con las especificaciones propuestas para los clientes de jWebSocket.	

Tabla 5: Descripción de la Tarea de Ingeniería 1.2 correspondiente a la HU_1.

Tarea de Ingeniería	
Número Tarea: 1.2	**Número Historia de Usuario:** HU_1
Nombre Tarea: Estudiar el protocolo de Hybi 13 para WebSocket.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 día
Fecha Inicio: 1/11/11	**Fecha Fin:** 1/11/11
Programador Responsable: Eylin Baydes González	
Descripción: La presente tarea tiene como objetivo implementar el cliente a bajo nivel que cumpla con las especificaciones del protocolo.	

Tabla 6: Descripción de la Tarea de Ingeniería 1.3 correspondiente a la HU_1.

Tarea de Ingeniería	
Número Tarea: 1.3	**Número Historia de Usuario:** HU_1
Nombre Tarea: Implementar el método handshake y generar clave.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 día
Fecha Inicio: 2/10/11	**Fecha Fin:** 2/11/11
Programador Responsable: Eylin Baydes González	
Descripción: Para establecer la conexión con el servidor el handshake juega un papel fundamental ya que está formado por un conjunto de reglas que deben ser seguidas por el cliente y el servidor. La clave es un elemento que no es estático y el cliente debe enviar una clave generada de manera aleatoria que cumpla con el estándar de UID. El servidor le agrega le añade una cadena acordada y le aplica el cifrado SHA1 y retorna la clave en la respuesta. El cliente realiza el mismo proceso y compara la clave enviada con la obtenida del servidor y solo si concuerdan queda establecida la conexión	

Tabla 7: Descripción de la Tarea de Ingeniería 1.4 correspondiente a la HU_1.

Tarea de Ingeniería	
Número Tarea: 1.4	**Número Historia de Usuario:** HU_1
Nombre Tarea: Implementar el método Open y las demás funcionalidades correspondiente con el método.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 día
Fecha Inicio: 3/11/11	**Fecha Fin:** 4/11/11
Programador Responsable: Eylin Baydes González	
Descripción: Esta tarea tiene como objetivo enviar el handshake, comparar la respuesta del servidor y establecer la conexión lanzando el evento on_open.	

Tabla 8: Descripción de la HU_2 enviar token al servidor.

Historia de Usuario	
Número: HU_2	**Nombre Historia de Usuario:** Enviar token al servidor.

Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Eylin Baydes González	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 6
Riesgo en Desarrollo: Alta	**Puntos Reales:** 6
Descripción: La presente historia de usuario tiene como objetivo enviar los tokens binarios o de texto hacia el servidor jWebSocket. Si estos exceden el tamaño máximo permitido se deben fragmentar y enviarlos como fragmentos independientes.	
Observaciones: Para ejecutar estas acciones debe haberse establecido la conexión previamente. Un token es el formato de datos propuesto por el servidor de jWebSocket para el intercambio de información.	
Prototipo de interface: Ninguna.	

Tabla 9: Descripción de la Tarea de Ingeniería 2.1 correspondiente a la HU_2.

Tarea de Ingeniería	
Número Tarea: 2.1	**Número Historia de Usuario:** HU_2
Nombre Tarea: Implementar funcionalidades para el envío de tipo texto al servidor.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2 semanas
Fecha Inicio: 7/11/11	**Fecha Fin:** 18/11/11
Programador Responsable: Eylin Baydes González.	
Descripción: Esta tarea tiene como objetivo enviar un mensaje al servidor. Para implementar el envío de texto se debe crear el un objeto de tipo raw_packet que describe a bajo nivel como está conformado el datagrama de información que se envía al servidor. Esta funcionalidad tiene su dificultad debido al enmascaramiento opcional de la información. Los datos son pasados por un proceso en el que se convierte la cadena a una lista de bytes para poder enmascarar y enviar al servidor.	

Tabla 10: Descripción de la Tarea de Ingeniería 2.2 correspondiente a la HU_2.

Tarea de Ingeniería	
Número Tarea: 2.2	**Número Historia de Usuario:** HU_2

Nombre Tarea: Implementar funcionalidades envío de binario al servidor.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2 semanas
Fecha Inicio: 21/11/11	**Fecha Fin:** 2/12/11
Programador Responsable: Eylin Baydes González.	
Descripción: La implementación se realiza para poder enviar un mensaje de tipo binario al servidor. Este método también requiere el enmascaramiento opcional de la información aunque se tiene la ventaja de no tener que convertir la cadena a arreglo de bytes.	

Tabla 11: Descripción de la Tarea de Ingeniería 2.3 correspondiente a la HU_2.

Tarea de Ingeniería	
Número Tarea: 2.3	**Número Historia de Usuario:** HU_2
Nombre Tarea: Implementar funcionalidades envío de fragmento al servidor.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2 semanas
Fecha Inicio: 5/12/11	**Fecha Fin:** 16/12/11
Programador Responsable: Eylin Baydes González.	
Descripción: Este método es usado puede ser usado de forma interna o externa y tiene como objetivo fragmentar un mensaje dado un tamaño máximo y enviarlos al servidor. El tamaño máximo del paquete debe ser especificado por parámetros de lo contrario se toma el tamaño máximo establecido por defecto en la librería cliente.	

Tabla 12: Descripción de la HU_3 recibir e interpretar token.

Historia de Usuario	
Número: HU_3	**Nombre Historia de Usuario:** Recibir e interpretar token.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Eylin Baydes González	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 4
Riesgo en Desarrollo: Alta	**Puntos Reales:** 4
Descripción: La presente historia de usuario tiene como objetivo recibir los tokens que envía el servidor y darles el tratamiento adecuado en dependencia de	

su tipo. Si son fragmentos se debe encargar de ensamblar su contenido.
Observaciones: Para ejecutar estas acciones debe haberse establecido la conexión previamente.
Prototipo de interface: Ninguna.

Tabla 13: Descripción de la Tarea de Ingeniería 3.1 correspondiente a la HU_3.

Tarea de Ingeniería	
Número Tarea: 3.1	**Número Historia de Usuario:** HU_3
Nombre Tarea: Implementar la funcionalidades de recibir token desde el servidor.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2 semanas
Fecha Inicio: 19/12/11	**Fecha Fin:** 14/01/12
Programador Responsable: Eylin Baydes González.	
Descripción: El objetivo que persigue esta tarea es recibir la información del servidor y notificarlo mediante el evento on_message. Un mensaje es recibido cuando en el buffer del socket se reciben bits de información. El mensaje es leído bit a bit por este método y siguiendo las especificaciones del datagrama que brinda el protocolo se va conformando el mensaje. Una vez que se obtiene el mensaje se crea una instancia de la clase token y se lanza en el evento on_message.	

Tabla 14: Descripción de la Tarea de Ingeniería 3.2 correspondiente a la HU_3.

Tarea de Ingeniería	
Número Tarea: 3.2	**Número Historia de Usuario:** HU_3
Nombre Tarea: Implementar la funcionalidades de recibir fragmento de token desde el servidor.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2 semanas
Fecha Inicio: 17/01/12	**Fecha Fin:** 28/01/12
Programador Responsable: Eylin Baydes González.	
Descripción: La funcionalidad leer fragmentos se trata de recibir un mensaje fragmentado y ensamblar el paquete completo conformando el objeto token como si nunca se hubiese tratado de un fragmento. Destacar que en este caso el	

tamaño máximo del fragmento no es especificado en el cliente si no del lado del servidor.

Tabla 15: Descripción de la HU_4 finalizar conexión con el servidor.

Historia de Usuario	
Número: HU_4	**Nombre Historia de Usuario:** Finalizar conexión con el servidor.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Eylin Baydes González	**Iteración Asignada:** 2
Prioridad en Negocio: Media	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio.	**Puntos Reales:** 1
Descripción: La presente historia de usuario tiene como objetivo finalizar la conexión con el servidor de WebSocket de acuerdo con la versión del protocolo seleccionado.	
Observaciones: Para ejecutar estas acciones debe haberse establecido la conexión previamente.	
Prototipo de interface: Ninguna.	

Tabla 16: Descripción de la Tarea de Ingeniería 4.1 correspondiente a la HU_4.

Tarea de Ingeniería	
Número Tarea: 4.1	**Número Historia de Usuario:** HU_4
Nombre Tarea: Implementar las funcionalidades para finalizar la conexión con el servidor.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 semana
Fecha Inicio: 31/01/12	**Fecha Fin:** 04/02/12
Programador Responsable: Eylin Baydes González.	
Descripción: Esta tarea de ingeniería tiene como objetivo finalizar la conexión con el servidor enviando el handshake de cierre de acuerdo con la versión del protocolo seleccionado. Cuando se cierra la conexión también se cierra el hilo utilizado para mantenerse a la escucha de los mensajes enviados desde el servidor de jWebSocket.	

2.6 Plan de Releases.

En este paso se define el plan de releases e iteraciones para realizar las entregas intermedias y la entrega final. 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 se llegó a la siguiente planificación:

Tabla 17: Plan de Release.

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
Iteración 2	En esta iteración se desarrolla la primera versión del producto en el cual se implementa las funcionalidades para establecer la conexión con el servidor, teniendo en cuenta la prioridad que representa para el desarrollo de la solución.	HU_1	1 semana
Iteración 3	En esta iteración se implementa las funcionalidades para el envío de información hacia el servidor, teniendo en cuenta la prioridad que representa para el desarrollo de la solución e integrándola con las funcionalidades de la iteración anterior.	HU_2 HU_3	10 semanas
Iteración 4	En esta iteración se implementa las funcionalidades para cerrar la conexión con el servidor, teniendo en cuenta la prioridad que representa para el desarrollo de la	HU_4	1 semana

	solución e integrándola con las funcionalidades de la iteración anterior.		

2.7 Descripción de la Arquitectura.

La arquitectura de las aplicaciones determina a gran escala como van a ser distribuidas las piezas del sistema, e influye incluso en la estructuración del código fuente. Aunque la librería cliente en python 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 otros componentes en otras capas a través de interfaces acordadas, donde cada capa agrega responsabilidad y abstracción a la capa directamente sobre ella.

La ventaja principal de una arquitectura distribuida en n capas es que el desarrollo se puede llevar a cabo en varios niveles y en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles. También se logra una mayor modularización de los componentes, lo que permite que estos puedan ser reutilizados en otros proyectos futuros.

Con el uso de una arquitectura en capas se ha logrado separar de forma estructural la librería cliente en dos partes fundamentales. Una capa para el cliente de bajo nivel y otra para el de alto. El cliente de bajo nivel realiza una completa implementación de las especificaciones del protocolo WebSocket. Este componente de bajo nivel no está ligado a la lógica del servidor de jWebSocket, solo se ajusta a los detalles inminentes de la conexión y la comunicación que establece el protocolo WebSocket. Esto posibilita su reutilización en cualquier proyecto desarrollado en Python que requiera una librería para conectarse a un servidor de WebSocket que cumpla con los estándares establecidos por este protocolo.

JWebSocket como marco de trabajo proporciona abstracciones y facilidades para la construcción de aplicaciones con este protocolo. La capa denominada cliente de

alto nivel ha sido implementada para suplir las abstracciones y lógica que propone el marco de trabajo jWebSocket en sus aplicaciones. Este componente difícilmente va a poder ser reutilizado para otro proyecto que no establezca las mismas normas o estándares para la comunicación que propone el marco de trabajo jWebSocket.

La librería ha sido dividida desde el punto de vista arquitectónico en las dos capas anteriormente mencionadas. Se desea esclarecer que el cliente de alto nivel depende enteramente del cliente de bajo nivel para manejar los detalles de la comunicación.

2.8 Diseño con Metáforas.

Debido a que SXP está basada en XP, y 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, el cual expone dicho diseño.

A continuación se representa el diagrama de paquetes para el sistema que se propone:

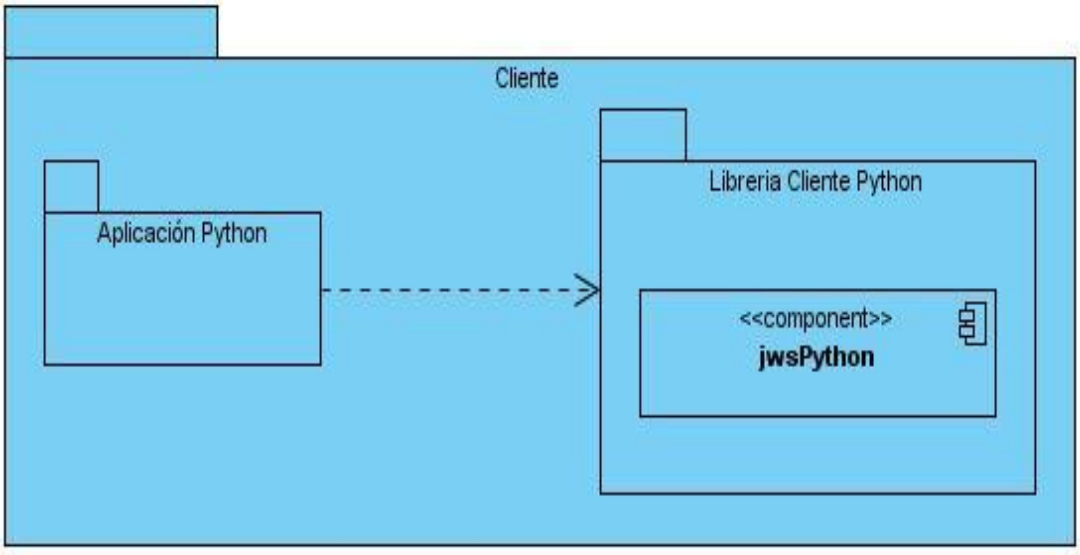


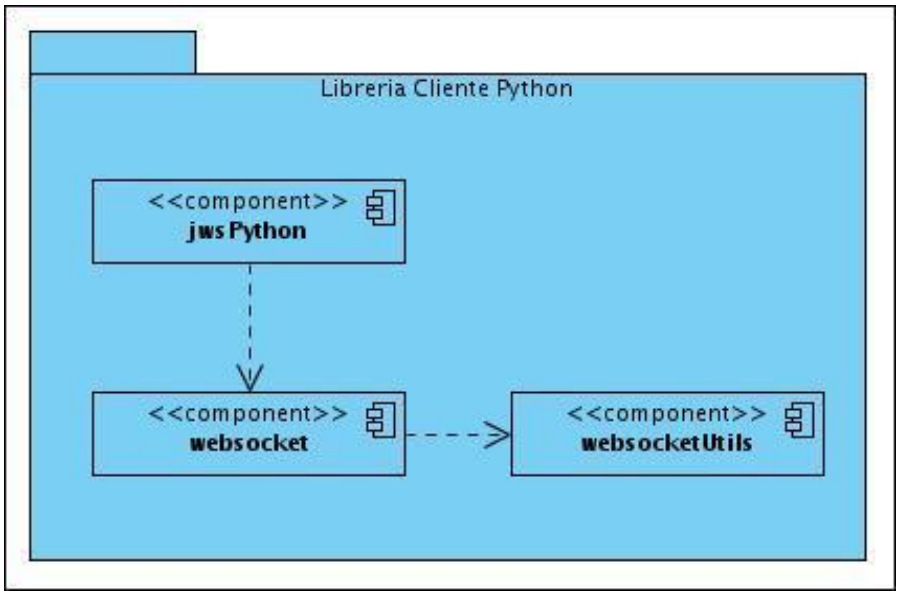
Figura 2: Diagrama de Paquete de la librería cliente python.

El paquete Cliente está compuesto por dos paquetes que simbolizan la estructura del directorio de la aplicación cliente, el paquete Aplicación Cliente representa la lógica de la aplicación, este paquete contiene todos los componentes concernidos con la aplicación cliente. El mismo tiene relación con el paquete librería cliente python el cual le permite la utilización de la API[26] y sus componentes para gestionar la conexión y comunicación con el marco de trabajo jWebSocket y el paquete librería cliente python 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 Componentes.

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.

A continuación se presenta el diagrama de componentes para el sistema que se propone:



[26] API: Interfaz *de* Programación *de* Aplicaciones.

Figura 3: Diagrama de Componentes de la librería cliente python.

En el diagrama se observan los siguientes componentes: jws Python es una capa de alto nivel que funciona como una abstracción para el protocolo WebSocket siguiendo las características y funcionalidades propias del servidor jWebSocket, WebSocket representa una implementación a bajo nivel de las especificaciones de la w3c y la IETF para el protocolo WebSocket en el lenguaje de programación Python, mediante este componente se puede establecer la comunicación con cualquier servidor de WebSocket que cumpla con los estándares definidos para el protocolo y WebSocketUtils contiene un conjunto de clases y algoritmos utilizados para la correcta implementación del protocolo.

Conclusiones del capítulo.

En el presente capítulo fueron definidas las características y principales funcionalidades de la librería cliente. De esta forma queda especificada la propuesta de solución del presente trabajo de diploma para la librería cliente en Python con el marco de trabajo jWebSocket. Además, se presentan los requisitos funcionales y no funcionales necesarios para la obtención de un módulo eficiente. Se describieron las historias de usuario y sus correspondientes tareas de ingeniería, así como el plan de release del proyecto definiéndose claramente el cronograma de trabajo y las tareas que el módulo debe realizar. Se realizó además, el modelado del diagrama de paquetes y de componentes que permiten un mejor entendimiento de la estructura de la librería cliente a desarrollar.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DEL SISTEMA.

Introducción.

En el presente capítulo se documenta la implementación de la solución propuesta en el capítulo anterior, generándose el diagrama de despliegue del sistema desarrollado. Además se exponen los casos de pruebas funcionales a los que fue sometida la aplicación en cada una de las iteraciones. El cumplimiento de estos casos de pruebas fue el hito para avanzar hacia la próxima iteración. En este capítulo además de las pruebas se dan a conocer los resultados obtenidos hasta el momento. Los aportes sociales y económicos a nivel mundial y en nuestro país.

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 se muestra el diagrama de despliegue, que representa la distribución física del sistema en términos de cómo se distribuirán las funcionalidades entre los nodos, donde cada nodo representa un recurso de cómputo, siendo estos procesadores o dispositivos hardware que se necesitan para el despliegue del sistema:

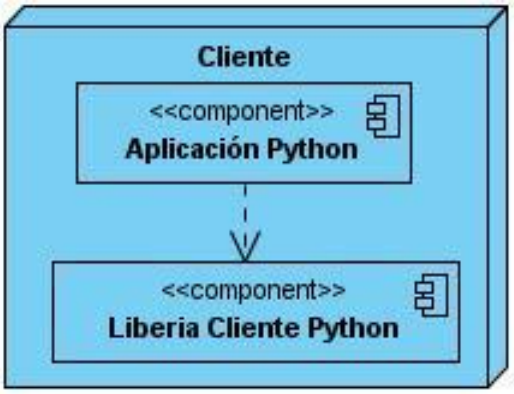


Figura 4: Diagrama de Despliegue del Sistema de la librería cliente python.

Las aplicaciones que son desarrolladas en Python que actúan como cliente requieren el uso de la librería para la comunicación con el servidor. Se plantea con comunicación todos los aspectos relacionados al manejo de la conexión y la entrada y salida de datos con el servidor. En el diagrama no se representa el

servidor de jWebSocket dado que esto no aporta nada a la solución, el servidor puede encontrarse en otro nodo físico en el mismo nodo o en clúster de estos y sigue siendo inherente a la librería cliente.

3.2 Validación de la investigación.

Para validar la presente investigación, que tiene como objetivo desarrollar una librería cliente que mejore los niveles de productividad y confiabilidad de las aplicaciones implementadas con Python y el marco de trabajo jWebSocket, se decidió trazar una estrategia de validación basada en las etapas siguientes:

En la primera etapa se realiza una aplicación demostrativa con el objetivo de poner a prueba el correcto funcionamiento de la librería cliente para python. El objetivo es certificar mediante esta, que la librería cumple con todos los requerimientos necesarios para su utilización. Después se hacen casos de pruebas funcionales alrededor de las historias de usuarios, que no son más que los requisitos funcionales que debe cumplir la aplicación diseñada.

Seguidamente se pasa a un proceso de certificación de calidad del software por el grupo de calidad del centro de desarrollo de la facultad regional Mártires de Artemisa, donde este grupo hace la certificación principalmente basándose en la funcionalidad, estandarización y limpieza del código. Este grupo también revisa los artefactos documentales que son generados por la metodología SXP que fue seleccionada para el desarrollo de esta investigación.

Por último, se somete la librería cliente desarrollada a una valoración por parte de Alexander Schulze, líder de la comunidad de jWebSocket internacional y arquitecto principal del proyecto. En esta etapa de validación se certifica la capacidad de integración de la solución desarrollada con el marco de trabajo jWebSocket a nivel internacional. A continuación se realiza una descripción más detallada de cada una de las etapas de la estrategia de validación trazada a las cuales fue sometida librería cliente desarrollada en python.

3.2.1 Aplicación demostrativa.

Con el objetivo de testear en una aplicación las funcionalidades de la librería, se ha desarrollado un pequeño sistema de prueba. Este sistema ha sido construido

utilizando la librería grafica PyQt. La aplicación permite básicamente las funcionalidades de un cliente de chat ligero. Para ejecutar la aplicación de prueba se ha implemento un plug-in en el servidor de jWebSocket, que actúa como servidor de chat. Este plug-in recibe los mensajes desde los clientes conectados y los reenvías hacia otros clientes. Se seleccionó realizar una aplicación de chat para probar las funcionalidades porque este tipo de sistemas cumplen con todos los requerimientos de una comunicación bidireccional y en tiempo real.

En una aplicación de chat los usuarios deben primero establecer la conexión con el servidor, lo que permite testear el primer requerimiento establecido para la librería cliente. Los usuarios pueden enviar y recibir mensajes, con esto se prueba la funcionalidad de enviar y recibir información al servidor. Por ultimo un usuario puede desconectarse en cualquier momento lo que permite evidenciar el cumplimiento de la funcionalidad cerrar la conexión con el servidor.

3.2.2 Casos de pruebas Funcionales.

Se denominan pruebas funcionales o Functional Testing, a las pruebas de software que tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados. Una prueba funcional es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software.

Posteriormente se dan a conocer las pruebas que se efectuaron a cada una de las historias de usuario de librería cliente python con el objetivo de comprobar el buen funcionamiento de la misma.

Como se ha venido explicando a lo largo de este documento la librería cliente de python para jWebSocket proporciona dos vías de utilización. Estas dos vías son el cliente a alto y bajo nivel. Con el objetivo de certificar lo anteriormente dicho e incrementar los niveles de calidad de ambos clientes, a cada historia de usuario se le realizó una prueba de aceptación dirigida a testear la funcionalidad en los clientes de alto y bajo nivel.

Tabla 18: Descripción del caso de prueba JWS-01-01 para la HU_1.

Caso de Prueba Funcionales

Código Caso de Prueba: JWS-01-01	**Nombre Historia de Usuario:** Establecer conexión con el servidor.
Nombre de la persona que realiza la prueba: Eylin Baydes González.	
Descripción de la Prueba: Esta prueba consiste en probar la funcionalidad establecer conexión con el servidor de jWebSocket, desde la óptica del cliente a bajo nivel. Para esto se instancia solo el cliente a bajo nivel desde una aplicación de consola y se ejecuta el método open, que inicializa la conexión con él envió del handshake correspondiente a la versión del protocolo. El servidor interpreta el handshake inicial y si responde de forma satisfactoria entonces la conexión queda establecida. Por disimiles motivos el servidor pude dejar de estar disponible en este caso se lanza una excepción que puede ser capturada y manejado a conveniencia del desarrollador. La excepción lanzada es del tipo WebSocketException.	
Condiciones de Ejecución: Para ejecutar esta prueba se requiere un ordenador con el intérprete de Python instalado y una consola para ejecutar el código. Se debe importar solo el componente WebSocket que forma parte de la librería. El servidor de jWebSocket debe estar corriendo bajo la URL especificada. Véase que no se requiere de ningún plug-in o evento en el servidor de jWebSocket para ejecutar esta prueba. Esto está dado por que el servidor solo responde al handshake de entrada incluso sin tener un servidor de tokens o listener activo. Otra condición importante que puede causar el no funcionamiento de la prueba es que el motor de TCP del servidor no este habilitado para recibir conexiones desde la maquina donde se ejecuta el cliente. Otro aspecto que debe ser verificado antes de ejecutar la prueba es el puerto por el cual está corriendo el servidor, por defecto el servidor de jWebSocket corre por el 8787, este puerto está dado por defecto en el cliente Python, si se desea seleccionar otro puerto debe entrarse en el patrón de url al inicializar la conexión.	
Entrada / Pasos de ejecución: El primer paso es echar a correr el servidor de jWebSocket, no resulta relevante si se corre desde un IDE o en modo solitario. Después solo resta ejecuta el código de Python que contiene la prueba desde la consola. En caso de que el servidor de jWebSocket este corriendo por un puerto	

que no sea el 8787, debe ser especificado en el patrón de url.
Resultado Esperado: El cliente a bajo nivel trata de establecer la conexión con el servidor de jWebSocket. En caso de establecerse la conexión en la consola se muestra el mensaje de bienvenida que envía el servidor de jWebSocket a todos los conectores una vez establecida la conexión de forma satisfactoria. En caso contrario se muestra el log causada por la excepción de tipo WebSocketException.
Evaluación de la Prueba: Satisfactoria.

Tabla 19: Descripción del caso de prueba JWS-01-02 para la HU_1.

Caso de Prueba Funcionales	
Código Caso de Prueba: JWS-01-02	**Nombre Historia de Usuario:** Establecer conexión con el servidor.
Nombre de la persona que realiza la prueba: Eylin Baydes González.	
Descripción de la Prueba: El objetivo de esta prueba es probar la funcionalidad establecer conexión con el servidor de jWebSocket. Esta vez desde el punto de vista del cliente de alto nivel. Para llevar a cabo la prueba correspondiente al cliente de alto nivel se implementó una aplicación con una interfaz gráfica basada en PyQt. El usuario selecciona la opción abrir conexión y la aplicación utiliza el cliente de alto nivel perteneciente a la librería para inicializar la conexión con el servidor de jWebSocket. Véase que aun tratándose del cliente a alto nivel tampoco se requiere de un plug-in específico para lograr la conexión por las mismas razones explicadas en la prueba anterior.	
Condiciones de Ejecución: Para ejecutar esta prueba se requiere un ordenador con el intérprete de Python instalado. Se debe importar la librería en su forma estándar, de esta forma se pueden utilizar todas las funcionalidades incorporadas en la librería. El servidor de jWebSocket debe estar corriendo bajo la URL especificada. Otra condición importante que puede causar el no funcionamiento de la prueba es que el motor de TCP del servidor no este habilitado para recibir conexiones desde la maquina donde se ejecuta el cliente.	

Entrada / Pasos de ejecución: El primer paso es echar a correr el servidor de jWebSocket, no resulta relevante si se corre desde un IDE o en modo solitario. Después el usuario de ir al menú de la aplicación y seleccionar la opción conexión y dentro de esta la opción de abrir conexión. Seguidamente se muestra una ventana en la cual se puede establecer la URL donde corre el servidor y hacer clic en el botón etiquetado con abrir conexión para establecer la conexión.
Resultado Esperado: En caso de establecer la conexión de forma satisfactoria se comprueba la funcionalidad del evento onOpen ya que mediante esta función que actúa como escuchador se adiciona el texto conexión establecida en un cuadro de texto. Si el servidor no se encuentra disponible se lanza un mensaje y se le da la posibilidad al usuario de reconectar con el servidor.
Evaluación de la Prueba: Satisfactoria.

Tabla 20: Descripción del caso de prueba JWS-02-01 para la HU_2.

Caso de Prueba Funcionales	
Código Caso de Prueba: JWS-02-01	**Nombre Historia de Usuario:** Enviar token al servidor.
Nombre de la persona que realiza la prueba: Eylin Baydes González.	
Descripción de la Prueba: El objetivo de esta prueba es enviar información al servidor. Se dice enviar información debido a que como se dijo anteriormente se prueban las funcionalidades en ambos clientes y en el cliente de bajo nivel no existe los tokens. A bajo nivel la información es enviada en tipo texto o binario. Esta información es empaquetada siguiendo las especificaciones del protocolo WebSocket. Una vez empaquetada la información se le da al paquete una estructura de notación de tipo JSON y se envía hacia el servidor por el socket TCP nativo de Python. Si la información es empaquetada correctamente según lo descrito en el protocolo WebSocket la información es interpretada sin ningún problema por el servidor de jWebSocket.	
Condiciones de Ejecución: Para ejecutar esta prueba se requiere un ordenador con el intérprete de Python instalado y una consola para ejecutar el código. Se	

debe importar solo el componente WebSocket que forma parte de la librería. Se hace necesario que el servidor este corriendo en una consola o IDE en el que se puedan visualizar los log del servidor y así ver como la información es recibida e interpretada correctamente por el servidor de jWebSocket. Se debe tener en cuenta que para enviar información al servidor la conexión debe haber sido previamente establecida.
Entrada / Pasos de ejecución: El primer paso es echar a correr el servidor de jWebSocket. Después se ejecuta el código Python que contiene la prueba desde la consola. Este script contiene el código que abre y envía un mensaje solicitado al servidor.
Resultado Esperado: Para verificar que el texto llega bien al servidor se estableció como precondición la utilización de una consola en la que aparece el mensaje recibido, el ejecutor de la prueba debe verificar los log del servidor y asegurarse de la autenticidad de los mensajes. Como el objetivo de la prueba es solo enviar información cliente no espera ninguna respuesta por parte del servidor solo envía y vuelve a preguntar por otro mensaje para enviar.
Evaluación de la Prueba: Satisfactoria.

Tabla 21: Descripción del caso de prueba JWS-02-02 para la HU_2.

Caso de Prueba Funcionales	
Código Caso de Prueba: JWS-02-02	**Nombre Historia de Usuario:** Enviar token al servidor.
Nombre de la persona que realiza la prueba: Eylin Baydes González.	
Descripción de la Prueba: El objetivo de esta prueba es enviar un token al servidor. Para la confección de esta prueba se ha desarrollado un Plug-in en el servidor de jWebSocket el cual recibe y reenvía la información recibida a todos los restantes conectores. Los tokens son la estructura de datos definida por el servidor de jWebSocket para lograr un mejor entendimiento en el intercambio de información. Para ejecutar esta prueba desde la aplicación cliente confeccionada con PyQt, se importa la librería completa ya que los tokens solo pueden ser	

usados por el componente de alto nivel. Una vez establecida la conexión el cliente cuenta con un cuadro de texto en el cual se introduce el mensaje para enviar al servidor. Con el mensaje se conforma un objeto de tipo Token que a su vez es empaquetado y codificado a JSON para luego ser enviado al servidor de jWebSocket. El token conformado posee un conjunto de funcionalidades que también son probadas como adicionarle campos y serializarlo como un objeto de tipo mapa.
Condiciones de Ejecución: Para ejecutar esta prueba se requiere un ordenador con el intérprete de Python instalado. Se debe ejecutar la aplicación grafica que permite ejecutar la prueba. Para enviar el token primero se requiere el establecimiento de la conexión por lo cual el usuario debe establecerla previamente. Se debe adicionar la configuración del plug-in cuyo nombre de espacio es (jws.py.demo) en el fichero de configuración del servidor de jWebSocket. También se debe crear una aplicación que importe el paquete java que contiene el código del plug-in mencionado para testear la funcionalidad de enviar token.
Entrada / Pasos de ejecución: El primer paso es echar a correr el servidor de jWebSocket. Ejecutar la aplicación con interface gráfica. Seleccionar la opción que estable la conexión. Introducir la información en el cuadro de texto de mensajes y presionar el botón enviar para conformar el token y enviarlo al servidor.
Resultado Esperado: Si el mensaje es enviado de forma satisfactoria se muestra en la lista de mensajes compartidos con otros conectores. La razón por la cual la información no llegaría al servidor es la perdida de conexión. En caso de que la causa resulte la perdida de la conexión se muestra un mensaje que indica se ha perdió la conexión. Si el servidor envía la información a otras conexiones entonces la prueba es satisfactoria por que pudo ser leído el token en el servidor.
Evaluación de la Prueba: Satisfactoria.

Tabla 22: Descripción del caso de prueba JWS-02-03 para la HU_2.

Caso de Prueba Funcionales	
Código Caso de Prueba: JWS-02-03	**Nombre Historia de Usuario:** Enviar token al servidor.
Nombre de la persona que realiza la prueba: Eylin Baydes González.	
Descripción de la Prueba: El objetivo de esta prueba consiste en enviar un token con un tamaño máximo establecido hacia el servidor jWebSocket. La información es introducida junto con el tamaño máximo de bytes que deben ser incorporados en cada fragmento del paquete. La información es dividida en varios paquetes y enviada en diferentes datagramas según se especifica en el protocolo WebSocket. Destacar que esta misma operación es realizada de forma anónima por la librería si la información que el usuario desea enviar sobrepasa el tamaño limite definido por el servidor de jWebSocket. La prueba es corrida desde una consola que ejecuta una aplicación Python que establece la conexión y envía una cadena de caracteres con un tamaño máximo definido.	
Condiciones de Ejecución: Para ejecutar esta prueba se ha realizado una porción de código Python que debe ser ejecutado sobre una consola. El servidor de jWebSocket debe estar corriendo y observar las traza para visualizar la información nuevamente devuelta a su forma original por el servidor de jWebSocket.	
Entrada / Pasos de ejecución: Se debe arrancar el servidor de servidor de jWebSocket. Para esta prueba tampoco se requiere de algún plugin adicionar en el servidor de jWebSocket. Se ejecuta el fichero que contiene la aplicación de Python desde una consola.	
Resultado Esperado: El cliente envía el mensaje contenido dentro del token en varios fragmentos hacia el servidor. En el servidor interpreta los paquetes y los ensamblas en uno solo siguiendo un procedimiento parecido al utilizado para empaquetar el paquete. Ver el paquete completo en la traza del servidor es la forma de validación que posee esta prueba.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 23: Descripción del caso de prueba JWS-03-01 para la HU_3.

Caso de Prueba Funcionales	
Código Caso de Prueba: JWS-03-01	**Nombre Historia de Usuario:** Recibir token desde el servidor.
Nombre de la persona que realiza la prueba: Eylin Baydes González.	
Descripción de la Prueba: El objetivo de esta prueba es recibir información al servidor. Se dice recibir información desde el servidor debido a que como se dijo anteriormente se prueban las funcionalidades en ambos clientes y en el cliente de bajo nivel no existe los tokens. A bajo nivel la información puede ser recibida en tipo texto o binario. Esta información es empaquetada siguiendo las especificaciones del protocolo WebSocket por parte del servidor. Para probar esta funcionalidad a bajo nivel se implementó una aplicación de consola que conectar con el servidor y permanece a la espera de mensajes provenientes desde el servidor. Los mensajes son recibidos como texto plano o como binario y son logueados directamente hacia la consola. En esta operación también se prueba el escuchador al evento de mensaje recibido para mostrar el mensaje.	
Condiciones de Ejecución: Para ejecutar esta prueba se requiere un ordenador con el intérprete de Python instalado y una consola para ejecutar el código. Se debe importar solo el componente WebSocket que forma parte de la librería. Se debe tener en cuenta que para recibir información desde el servidor la conexión debe haber sido previamente establecida.	
Entrada / Pasos de ejecución: El primer paso es echar a correr el servidor de jWebSocket. Después se ejecuta el código Python que contiene la prueba desde la consola. Este script contiene el código que abre y envía un mensaje solicitado al servidor.	
Resultado Esperado: El resultado esperado es que el cliente sea capaz de recibir e interpretar todos los mensajes provenientes desde el servidor. Una vez recibido el mensaje el cliente debe ejecutar el evento on_message con el que se loguea el resultado en la consola.	

Evaluación de la Prueba: Satisfactoria.

Tabla 24: Descripción del caso de prueba JWS-03-02 para la HU_3.

Caso de Prueba Funcionales	
Código Caso de Prueba: JWS-03-02	**Nombre Historia de Usuario:** Recibir token desde el servidor.
Nombre de la persona que realiza la prueba: Eylin Baydes González.	
Descripción de la Prueba: El objetivo de esta prueba es recibir la información proveniente del servidor y confeccionar el objeto de tipo token. Para la confección de esta prueba se ha desarrollado un Plug-in en el servidor de jWebSocket el cual recibe y reenvía la información recibida a todos los restantes conectores. El usuario puede conectar varias aplicaciones clientes y enviar token desde unas hacia otras. Los tokens son la estructura de datos definida por el servidor de jWebSocket para lograr un mejor entendimiento en el intercambio de información. Para ejecutar esta prueba desde la aplicación cliente confeccionada con PyQt, se importa la librería completa ya que los tokens solo pueden ser usados por el componente de alto nivel. Una vez establecida la conexión el cliente comienza a escuchar por la llegada de un token desde el servidor. Los token son recibidos cuando otros clientes envían información e impresos en un cuadro de texto en el cual aparece toda la información recibida y enviada.	
Condiciones de Ejecución: Para ejecutar esta prueba se requiere un ordenador con el intérprete de Python instalado. Se debe ejecutar la aplicación grafica que permite ejecutar la prueba. Para enviar el token primero se requiere el establecimiento de la conexión por lo cual el usuario debe establecerla previamente. Se debe adicionar la configuración del plug-in cuyo nombre de espacio es (jws.py.demo) en el fichero de configuración del servidor de jWebSocket. También se debe crear una aplicación que importe el paquete java que contiene el código del plug-in mencionado para testear la funcionalidad de enviar token.	
Entrada / Pasos de ejecución: El primer paso es echar a correr el servidor de jWebSocket. Ejecutar la aplicación con interface gráfica. Seleccionar la opción	

que estable la conexión.
Resultado Esperado: Cuando un mensaje es recibido de forma satisfactoria desde el servidor se muestra el texto dentro de cuadro de texto de mensajes compartidos entre conectores. La razón por la cual la información no llegaría desde el servidor es la perdida de conexión. Si el cliente recibe la información a proveniente desde el servidor entonces la prueba es satisfactoria por que pudo ser interpretar los el tokens recibidos.
Evaluación de la Prueba: Satisfactoria.

Tabla 25: Descripción del caso de prueba JWS-04-01 para la HU_4.

Caso de Prueba Funcionales	
Código Caso de Prueba: JWS-04-01	**Nombre Historia de Usuario:** Finalizar Conexión.
Nombre de la persona que realiza la prueba: Eylin Baydes González.	
Descripción de la Prueba: La siguiente prueba tiene como objetivo finalizar la conexión entre la aplicación cliente y el servidor jWebSocket por parte del cliente a bajo nivel. Para llevar a cabo esta prueba el cliente conforma un mensaje con los datos necesarios para finalizar la conexión conocido como handshake de cierre, descritos en el protocolo WebSocket. Este paquete es conformado siguiendo las especificaciones del protocolo y finalmente enviado hacia el servidor, el cual le dará un tratamiento diferente que al resto de los paquetes que recibe. El servidor procesa este como un paquete de cierre y comienza a desconectar el conector asociado al mensaje de cierre. El servidor es para de enviar envió de paquetes de tipo Pong y envía como respuesta a la solicitud del cliente un paquete 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 él envió de paquetes Ping y cierra el socket TCP que albergaba la conexión.	

Condiciones de Ejecución: Para ejecutar esta prueba el cliente y el servidor de jWebSocket deben estar previamente conectados. Para probar el cliente de bajo nivel se implementó una aplicación de consola que debe ser ejecutada.
Entrada / Pasos de ejecución: Se inicializa el servidor de jWebSocket. Después se ejecuta la aplicación de consola que de forma automática establece la conexión con el servidor y pide al usuario que presione una tecla para finalizar la conexión.
Resultado Esperado: El cliente debe finalizar la conexión ejecutando el evento on_close el cual se utiliza para mostrar un mensaje de cierre de conexión.
Evaluación de la Prueba: Satisfactoria.

Tabla 26: Descripción del caso de prueba JWS-04-02 para la HU_4.

Caso de Prueba Funcionales	
Código Caso de Prueba: JWS-04-02	**Nombre Historia de Usuario:** Finalizar Conexión.
Nombre de la persona que realiza la prueba: Eylin Baydes González.	
Descripción de la Prueba: La siguiente prueba tiene como objetivo finalizar la conexión entre la aplicación cliente y el servidor jWebSocket por parte del cliente a alto nivel. Para llevar a cabo esta prueba se utiliza la aplicación de interfaz gráfica con PyQt que utiliza el cliente de alto nivel. El usuario debe seleccionar la opción cerrar conexión en el menú de conexión. Con esto se envía el handshake de cierre al servidor.	
Condiciones de Ejecución: Para ejecutar esta prueba el cliente y el servidor de jWebSocket deben estar previamente conectados. Se debe ejecutar la aplicación de interfaz gráfica y establecer la conexión.	
Entrada / Pasos de ejecución: Se inicializa el servidor de jWebSocket. Después se ejecuta la aplicación de interfaz gráfica y se conecta con el servidor. Para	

finalizar la prueba se selecciona la opción cerrar conexión.
Resultado Esperado: Se espera el cierre de la conexión de forma tal que no se recibirán más mensajes desde otros conectores y se ejecutara el evento on_close el cual es usado para mostrar un mensaje de confirmación de cierre de conexión.
Evaluación de la Prueba: Satisfactoria.

3.2.3 Certificación de Calidad de Software.

El grupo de calidad del Centro de Desarrollo de la Facultad Regional "Mártires de Artemisa" es el encargado de realizar el proceso de certificación de calidad del código fuente desarrollado, así como de la documentación generada teniendo en cuenta la metodología de desarrollo SXP seleccionada en la presente investigación. La revisión del código fuente es llevada a cabo por el Ing. Domma Moreno Dager, Asesor de Tecnología del Centro. Durante esta revisión se evalúa la estandarización del código fuente basándose principalmente en la Plantilla Estándar de Código generada con este propósito. El hecho de que la librería cliente desarrollada cumpla estrictamente con los estándares definidos para el proyecto facilita el mantenimiento posterior del código, garantizando a su vez la calidad y limpieza del mismo. Por otra parte, durante el proceso de revisión del código fuente, se analiza el funcionamiento de la librería cliente desarrollada teniendo en cuenta los requisitos funcionales definidos por el cliente. Con este objetivo se ejecutan un conjunto de pruebas funcionales basadas en las Historias de Usuarios definidas para esta librería. Estas pruebas funcionales permiten verificar la calidad y el correcto funcionamiento de la solución.

El proceso de revisión de la documentación generada para la librería cliente desarrollada es llevado a cabo por la Ing. Maidel Ojeda Castro, Asesora de Calidad del Centro de Desarrollo. La documentación a evaluar está compuesta por los siguientes artefactos:

- ✓ 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 Manual de Usuario
- ✓ Plantilla Manual de Administración
- ✓ Plantilla Manual de Desarrollo

Estos artefactos se generan teniendo en cuenta la metodología SXP y describen los principales aspectos del proceso de desarrollo de software de la librería cliente. Durante el proceso de revisión de esta documentación se evalúa el cumplimiento con las plantillas definidas por la metodología SXP para cada uno de estos artefactos. Además se verifica la claridad en la redacción de toda la documentación de forma tal que facilite el entendimiento de la misma por parte de otros desarrolladores o usuarios. A su vez estos artefactos deben cumplir con los estándares de calidad establecidos por la Universidad de la Ciencias Informáticas para garantizar de esta forma la continuidad del proyecto.

3.2.4 Valoración del Cliente.

La librería cliente python desarrollada en la presente investigación fue sometido al criterio de Alexander Schulze, fundador y principal arquitecto del proyecto jWebSocket, así como líder de la comunidad internacional del mismo. Para su evaluación se almacena en el repositorio internacional de jWebSocket el código fuente de la librería desarrollada y una documentación en inglés que posibilita un mayor entendimiento de la solución. El análisis del código fuente tiene como objetivo principal verificar la capacidad de utilización en una aplicación cliente con la librería cliente python y el marco de trabajo. Con este fin se evalúa la limpieza del código fuente y los estándares de codificación establecidos para este marco de trabajo. De esta forma se permita incrementar la calidad de la solución así como su correcto funcionamiento en correspondencia con jWebSocket.

La documentación en inglés asociada a la librería desarrollada está compuesta por los manuales de Usuario, Desarrollador y Administración. Esta documentación no solo es utilizada para lograr un mayor entendimiento de la solución sino que a su vez es evaluada exhaustivamente con el objetivo de incrementar su calidad. De esta forma estos manuales pueden ser usados para facilitar el soporte de la librería desarrollada una vez que esta sea liberada e integrada al marco de trabajo jWebSocket.

3.3 Resultados Obtenidos.

El proceso de validación al cual fue sometida la librería cliente python desarrollada, arrojó los siguientes resultados:

- ✓ Los casos de pruebas funcionales realizados fueron satisfactorios, permitiendo el correcto funcionamiento de la librería cliente python desarrollada y el cumplimiento de los requisitos funcionales definidos por el cliente.
- ✓ El grupo de calidad del Centro de Desarrollo de la Facultad Regional “Mártires de Artemisa” emitió un aval que certifica la funcionalidad y estandarización de la librería cliente python desarrollada, así como la calidad de la documentación que permite a la solución ser usable y generalizada para otros usuarios.
- ✓ La valoración por parte de Alexander Schulze certificó satisfactoriamente la librería cliente python desarrollada, asegurando de esta forma el cumplimiento de los requerimientos presentados por el marco de trabajo jWebSocket, así como su usabilidad real por los clientes potenciales y desarrolladores de esta comunidad.

Teniendo en cuenta los resultados satisfactorios de cada una de las etapas de la estrategia de validación asumida en la presente investigación, queda disponible la versión 1.0 de la librería cliente python. Se obtuvo una Librería que cumple con todas especificaciones del cliente para la comunicación entre una aplicación cliente y el servidor de jWebSocket.

3.4 Funcionalidades Obtenidas.

Entre las principales funcionalidades que posee la librería cliente python en su versión 1.0 se pueden nombrar:

- ✓ **Establecer la conexión con el servidor de jWebSocket:**
  Esta funcionalidad consiente en realizar la conexión entre una aplicación desarrollada con el lenguaje de programación Python y el servidor de jWebSocket.
- ✓ **Enviar token de tipo texto hacia el servidor de jWebSocket:**
  Esta funcionalidad consiente en enviar la información desde la aplicación cliente hacia el servidor de jWebSocket utilizando una estructura de datos abstracta llamada token, y el tipo de datos que puede enviar es texto.
- ✓ **Enviar token de tipo binario hacia el servidor de jWebSocket:**
  Esta funcionalidad consiente en enviar la información desde la aplicación cliente hacia el servidor de jWebSocket utilizando una estructura de datos abstracta llamada token, y el tipo de datos que puede enviar es binario.
- ✓ **Enviar token fragmentado hacia el servidor de jWebSocket:**
  Esta funcionalidad consiente en enviar información desde la aplicación cliente hacia el servidor de jWebSocket utilizando la estructura de datos abstracta token. Si la información enviada por el cliente 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 consiente en recibir la información procedente del servidor jWebSocket y procesarla para empaquetarla utilizando el tipo de dato abstracto token. Luego de empaquetarla se le brindará la información a la aplicación cliente que utiliza esta librería cliente.
- ✓ **Recibir e Interpretar fragmentos de token entrantes desde el servidor de jWebSocket:**
  Esta funcionalidad consiente en recibir la información procedente del servidor de jWebSocket y procesarla para empaquetarla utilizando el tipo de dato abstracto token. Cuando la información se encuentra ya fragmentada,

entonces se emplea un poderoso algoritmo para ensamblarla y finalmente ofrecérsela a la aplicación cliente que utiliza esta librería cliente.

- ✓ **Cerrar la conexión con el servidor de jWebSocket:**
  Esta funcionalidad consiente en enviarle un token con la información de cierre al servidor de jWebSocket, el cual le responderá esa petición y suspenderá dicha conexión.

3.5 Aporte Social y Económico.

La librería cliente python desarrollada para trabajar con el marco de trabajo jWebSocket representa una alternativa viable para el desarrollo de aplicaciones clientes en tiempo real. Dicha librería permite una completa integración entra las aplicaciones clientes y el servidor de jWebSocket. Brinda un considerable ahorro de tiempo y recursos a las empresas y equipos de desarrollos que utilizan el lenguaje de programación Python para utilizar sistemas desarrollados con el marco de trabajo jWebSocket. Además de permitir a los desarrolladores centrarse en la lógica de la aplicación en vez de tratar de implementar el protocolo WebSocket para poder establecer la conexión con el servidor jWebSocket.

Las ventajas anteriormente mencionadas constituyen un impacto económico ya que cualquier desarrollador o empresa puede utilizar la librería para desarrollo de aplicaciones clientes. La universidad que se presenta como autor de esta librería también puede asumir las ganancias relacionadas al soporte y la capacitación de los desarrolladores.

En la sociedad también se evidencia el impacto ya que licencia LGPL la hace disponible gratuitamente para cualquier desarrollador que use el lenguaje de programación Python. Dicha librería ha sido desarrollada en la Universidad de las Ciencias Informáticas y por tanto la se beneficia del prestigio que esto aporta. La librería también puede ser utilizada como material de estudio en la capacitación de estudiantes, profesores y trabajadores de esta universidad. Esto representa un gran aporte a la comunidad de desarrolladores de dicha institución.

Conclusiones del Capítulo.

En este capítulo se realizó el diagrama de despliegue que indica la situación física de los componentes lógicos desarrollados. Además, fue establecida una estrategia

de validación que permitió certificar el correcto funcionamiento del módulo desarrollado, demostrando que sus funcionalidades satisfacen las necesidades del cliente y concuerdan con los requisitos definidos en la etapa inicial del proyecto. De esta manera se realizaron pruebas de aceptación a la librería cliente, las cuales permitieron demostrar el excelente funcionamiento de la misma. También se dio a conocer los aportes sociales y económicos que aporta la librería. Como resultado final de la presente investigación se obtuvo la versión 1.0 de la librería cliente en python la cual cuenta con las funcionalidades necesarias para incrementar el correcto funcionamiento de la misma.

CONCLUSIONES GENERALES

Con la realización del presente trabajo de diploma se dio respuesta a cada una de las preguntas científicas planteadas, obteniéndose de manera general las siguientes conclusiones:

- ✓ Se realizó la conceptualización de la comunicación mediante el protocolo WebSocket en las librerías clientes.
- ✓ Actualmente existen librerías clientes en python que se comunican mediante el protocolo WebSocket, pero no para el marco de trabajo jWebSocket.
- ✓ La librería cliente python posibilita el desarrollo de aplicaciones en tiempo real utilizando el marco de trabajo jWebSocket.
- ✓ Se obtiene un posible aumento del nivel de productividad y confiabilidad para las aplicaciones clientes desarrolladas utilizando la librería cliente python y el marco de jWebSocket.

RECOMENDACIONES

Se recomienda para versiones posteriores de la librería cliente python:

- ✓ Incluir en la próxima versión de la librería cliente python 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.

REFERENCIA BIBLIOGRÁFICA

Ben-Ari, M. 1990. *"Principles of Concurrent and Distributed Programming".* 1990. ISBN 0-13-711821-X.

Collins-Sussman, Ben, W. Fitzpatrick, Brian and Michael Pilato, C. 2004. *Version Control with Subversion.* USA. : O'Reilly, 2004.

Dayley, Brad. 2007. *Python Phrasebook.* 2007.

Figueroa, Roberth G., Solís , Camilo J. and Cabrera, Armando A. 2010. *Metodología Tradicionales vs. Metodología Ágiles.* 2010.

González Duque, Raúl. 2010. *Python para todos.* España : s.n., 2010.

Hybi. 2011. The WebSocket protocol. *The WebSocket protocol.* [Online] septiembre 30, 2011. [Cited: diciembre 1, 2011.] http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-17.

INTECO. 2009. *METODOLOGÍAS Y CICLOSDE VIDA.* 2009.

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.

Klew, Willy. 2011. *Visua Beta.* [Online] Geany, 2011. [Cited: 02 19, 2012.] http://www.visualbeta.es/22227/software/geany-un-ide-multiplataforma-liviano-y-muy-completo/.

Menéndez, Evelyn. 2011. Plusformacion.com. [Online] 2011. [Cited: 02 19, 2012.] http://www.plusformacion.com/Recursos/r/Herramientas-CASE-para-procesodesarrollo-Software?quicktabs_ofertas_relacionadas_quicktab=1#resumena..

Menéndez, Rosa. 2011. *Proyectos.* [Online] 03 03, 2011. [Cited: 02 20, 2012.] http://www.usmp.edu.pe/publicaciones/boletin/fia/info36/proyectos.html#a..

Módulo para la realización de presentaciónes web reusable sobre Moodle. **2011.** s.l. : Autoedición, 2011.

omercade. 2010. Omitsis. [Online] 03 07, 2010. [Cited: 02 19, 2012.] http://www.omitsis.com/scrum-como-metodologia-de-desarrollo.

Paez , Andrea. 2011. SAD. [Online] 04 02, 2011. [Cited: 02 21, 2012.] http://mistock.lcompras.biz/paradigmas23/941-git-software-de-control-de-versiones.

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

Pérez Bisset, Clenda and Mojena Fis, Kenia María . 2009. *Herramienta para la administración de repositorios Subversion (PhpSvnAdmin).* La Habana : s.n., 2009. p. http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_2097_09, Tesis de grado.

Schmuller, Joseph. 2010. *Aprendiendo UML en 24 Horas.* Mexico : Division Computacion, 2010.

Schulze, A. (2011). 2011. *Framework Approach for WebSockets.* 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, 2011.

Sybase. 2009. *Client-Library Migration Guide.* 2009. p. http://www.sybase.com/detail?id=1011207.

The Linux Information Project. 2005. The Linux Information Project. [Online] 11 5, 2005. [Cited: 02 25, 2012.] http://www.linfo.org/full_duplex.html.

VARGAS, P. A. 2011. [Online] 2011. http://repositorio.espe.edu.ec/bitstream/21000/678/1/T-ESPE-014103.pdf.

Visual Paradigm International Ltd. 2011. Visual Paradigm for UML. *Visual Paradigm.* [Online] 2011. [Cited: 02 20, 2012.] http://www.visual-paradigm.com/product/vpuml/.