Universidad de las ciencias Informáticas
Facultad Regional Mártires de Artemisa

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

Título:

Tablero de cotizaciones en un entorno de tiempo real usando el marco de trabajo jWebSocket.

Autor:
Roylandi González Pujol

Tutor:
MsC. Yamila Vigil Regalado

Cotutor:

Dr. Raúl R. Crespo Heredia

Artemisa, Junio 2012

DECLARACÍON DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo al <nombreárea> 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 _____.

Roylandi González Pujol **Msc. Yamila Vigil Regalado**

_____ _____

Firma del Autor Firma del Tutor

Dr. Raúl R. Crespo Heredia

_____

Firma del Cotutor

Agradecimientos

Agradezco a mis padres, por la confianza que han depositado en mí a lo largo de estos años, por su esfuerzo y dedicación, que ha hecho de mí el hombre que soy hoy.

Agradezco a mi familia por su apoyo incondicional principalmente a mis abuelos Mami y Papi que fueron mis segundos padres, a mis hermanos Rony, Andy y Martincito que siempre están con migo y a mis tías Cary y Pupy.

Agradezco a mis tutores por toda su ayuda y orientación durante la realización de esta investigación.

Agradezco a todos los que de una forma u otra influyeron en todo mi tiempo de estudio en mi formación como un buen profesional.

Agradezco a mis amigos del cuarto especialmente a Feyt, Tito, Jeikel, y El Riki que han estado a mi lado en todos los momentos buenos y malos a lo largo de estos 5 años. A mi amiga Merly por su ayuda incondicional.

Agradezco a Anita mi amiga y compañera, por su apoyo incondicional, por creer en mí y por ayudarme a superar muchos obstáculos que sin ella no hubieran sido posibles.

En fin agradezco a todo aquel que ha hecho posible la realización de mi sueño.

Dedicatoria

Dedico este trabajo de Diploma a mis padres, por creer más en mí de lo que yo mismo he creído, por su incondicional e infinito amor. Siempre han sido mi guía y ejemplo ante la vida. Gracias por ser los mejores padres del mundo.

A mis segundos padres Mami y Papi, por su entrega incondicional, por cuidarme de donde quiera que estén, por siempre están con migo.

A mi hermano Rony por transmitirme sus fuerzas y darme su cariño y apoyo día a día.

A mi familia por apoyarme en todo momento.

A todos mis amigos.

RESUMEN

Un tablero de cotizaciones es una herramienta que permite reflejar valores financieros brindando las facilidades necesarias para que las empresas introduzcan órdenes de compra y venta de valores. En la actualidad su uso principal es en la bolsa de valores donde las empresas ponen a la venta sus acciones dando a conocer el incremento y decremento del valor de las mismas.

La Web desde su creación ha ido evolucionando a través del desarrollo de nuevas tecnologías. En este ámbito surge el protocolo WebSocket el cual permite establecer una comunicación bidireccional entre el cliente y el servidor a través de un *socket* TCP garantizando tiempo real en las aplicaciones web. Las aplicaciones que representan tableros de cotizaciones en la Web se han desarrollado de acuerdo con el desarrollo. Hoy día se puede consultar tableros de cotizaciones web con el uso del protocolo WebSocket.

jWebSocket es un marco de trabajo para el desarrollo de aplicaciones estacionarias y móviles mediante el protocolo WebSocket. Sin embargo el marco de trabajo no cuenta hoy con una vía directa que muestre sus potencialidades en el desarrollo de tableros de cotizaciones en la Web en tiempo real.

La presente investigación tiene como objetivo desarrollar una aplicación web demostrativa para la gestión de datos mediante un tablero de cotizaciones de tiempo real con jWebSocket. La aplicación constituye así una vía directa de mostrar la capacidad del marco de trabajo en el desarrollo aplicaciones financieras en la Web garantizando mayor nivel de usabilidad del marco en la comunidad internacional.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las empresas transnacionales nacieron debido a las consecuencias del proceso de ampliación de los mercados, teniendo como característica extenderse por todo el mundo. Con el transcurso de los años el crecimiento acelerado de estas empresas ha ganado gran influencia en la sociedad debido a su constante evolución por las grandes exigencias del entorno empresarial. El sector empresarial requiere alta competitividad puesto que constituye un marco donde son altos los riesgos, apostando todo a decisiones que definen el futuro de las empresas.

A comienzos del siglo XX se cuentan con miles de empresas multinacionales, proceso que sigue incrementándose de manera exponencial. A finales de este siglo en el sector empresarial no existía un auge del uso de las Tecnologías de la Información y las Comunicaciones (TIC). Sin embargo este sector se ve beneficiado por la suspensión de las prohibiciones del comercio internacional y la apertura de la inversión extranjera en muchos países.

El surgimiento de la Web, la evolución y el desarrollo de las TIC ha dado paso a su uso por las empresas transnacionales. Este hecho ha permitido extender la compra y venta de productos y servicios del mercado tradicional a lo que hoy se conoce comercio electrónico. Este medio por su flexibilidad cada día suma más usuarios en línea exigiendo la creación de aplicaciones más potentes que satisfagan sus necesidades.

Las aplicaciones en la Web han extendido su uso desde el comercio electrónico hasta los tableros de cotizaciones que visualizan datos financieros. Estos últimos son instrumentos donde las empresas reflejan sus finanzas brindando las facilidades necesarias para que sus miembros, atendiendo las disposiciones de sus clientes, introduzcan órdenes y realicen negociaciones de compra y venta de valores.

Un tablero de cotizaciones no es más que una aplicación donde la información de las tablas varía de forma dinámica en el tiempo. En la actualidad su principal uso es

en bolsas de valores, donde las empresas ponen a la venta sus acciones dando a conocer el incremento y decremento del valor de las mismas en función del tiempo. Esto permite a las personas convertirse en socios o crear sus negocios propios sin la necesidad de tener un gran capital. Se utiliza además para mostrar cambios en el valor de las monedas, de los recursos minerales, entre otros.

Dado que en los tableros de cotizaciones se interactúa con datos financieros reales, definiendo el destino del capital de las empresas o de los usuarios, se hace necesario que esta información fluya en tiempo real. Tanto los miembros como los clientes desean recibir retroalimentación inmediata de todas las acciones que se realizan. Ante las nuevas necesidades de los usuarios de la Web en lograr tiempo real en la actualización de la información surge el protocolo de comunicación WebSocket. Este protocolo a diferencia de su precedente HTTP no sigue un esquema solicitud-respuesta, garantizando una comunicación bidireccional y *full-duplex* entre el cliente y el servidor web.

WebSocket hoy día constituye un cambio de paradigma para el desarrollo de aplicaciones web estacionarias y móviles. La forma de comunicación entre el cliente y el servidor sin una previa solicitud del usuario hace que la información sea inmediata logrando tiempos reales en la Web.

La visualización de datos de un tablero de cotizaciones en tiempo real garantiza una mayor transferencia de información. Hoy en día para desarrollar un tablero de cotizaciones en tiempo real existen tecnologías robustas que utilizan el protocolo WebSocket. Algunos de los principales servidores que soportan este protocolo son la pasarela Websockets de Kaazing[1], JettyWebSocketServlet[2], Socket.IO[3], django-websocket del proyecto Python[4] y jWebsocket[5].

---

[1]http://kaazing.com/

[2]http://www.eclipse.org/jetty/

[3]http://socket.io/son

[4]http://pypi.python.org/pypi/django-websocket

jWebSocket es un marco de trabajo de código abierto que proporciona altos niveles de seguridad y escalabilidad para el desarrollo de aplicaciones en tiempo real para la Web, usando el protocolo WebSocket. Este permite implementar las aplicaciones del lado del servidor utilizando el lenguaje de programación multiplataforma Java y del lado del cliente JavaScript. Hoy el marco cuenta con un núcleo en Java y clientes en Java y en JavaScript que permiten el desarrollo de aplicaciones que gestionan datos financieros en la Web, entre ellos tableros de cotizaciones. Sin embargo el proyecto jWebSocket no cuenta con una vía directa que muestre a la comunidad sus potencialidades para desarrollar tableros de cotizaciones en la Web en tiempo real.

A raíz de lo anteriormente planteado surge la siguiente **situación problemática**:

En la actualidad no existe una vía que muestre la capacidad del marco de trabajo jWebSocket para desarrollar aplicaciones Web que gestionen datos mediante un tablero de cotizaciones en tiempo real. Dado estos hechos los desarrolladores no identifican con claridad las potencialidades que ofrece este marco de trabajo para gestionar datos en un tablero de cotizaciones. Esto trae consigo que múltiples empresas que desarrollan aplicaciones de tableros de cotizaciones en la Web, no reconozcan el uso de jWebSocket como una vía eficiente para lograr este fin. De esta manera no se logran altos niveles de usabilidad del marco jWebSocket en el desarrollo de aplicaciones web de tableros de cotizaciones. El bajo nivel de conocimiento de las potencialidades de jWebsocket para la gestión de datos mediante un tablero de cotizaciones en tiempo real trae consigo además pérdidas de cuotas de mercado para dicho marco de trabajo.

De lo planteado anteriormente nace la siguiente interrogante que nos define el **Problema Científico**:

¿Cómo garantizar mayor nivel de usabilidad del marco de trabajo jWebSocket en el desarrollo de aplicaciones que gestionen datos mediante un tablero de cotizaciones

---

[5]https://jwebsocket.org/

de tiempo real en la Web?

Teniendo en cuenta todo lo anterior se define como **Objeto de estudio:**

- ✓ Gestión de datos mediante un tablero de cotizaciones de tiempo real en la Web.

Dentro del objeto de estudio anterior se define como **Campo de Acción:**

- ✓ Gestión de datos mediante un tablero de cotizaciones de tiempo real en la Web con el marco de trabajo jWebSocket.

Para dar solución al problema antes expuesto, se plantea como **Objetivo general:**

Desarrollar una aplicación web para la gestión de datos mediante un tablero de cotizaciones de tiempo real con jWebSocket que garantice mayor nivel de usabilidad del marco de trabajo en el desarrollo de aplicaciones financieras en la Web.

Para dar cumplimiento al objetivo general planteado se han derivado los siguientes **Preguntas Científicas:**

- ✓ ¿Cuáles son los fundamentos teórico-metodológicos de la gestión de datos mediante un tablero de cotizaciones de tiempo real en la Web?
- ✓ ¿Cuál es la situación actual de la gestión de datos mediante un tablero de cotizaciones de tiempo real en la Web?
- ✓ ¿Cómo desarrollar una aplicación web demostrativa para la gestión de datos mediante un tablero de cotizaciones de tiempo real con jWebSocket que garantice mayor nivel de usabilidad del marco de trabajo en el desarrollo de aplicaciones financieras en la Web?
- ✓ ¿Cuáles son los resultados obtenidos al utilizar la aplicación web demostrativa desarrollada para garantizar mayor nivel de usabilidad del marco de trabajo jWebSocket en el desarrollo de aplicaciones financieras en la Web?

Para dar cumplimiento a las preguntas científicas propuestas se planificaron las siguientes **Tareas de la Investigación:**

1. Fundamentación teórico-metodológica de la gestión de datos mediante un tablero de cotizaciones de tiempo real en la Web.
2. Caracterización de la situación actual de la gestión de datos mediante un tablero de cotizaciones de tiempo real en la Web.
3. Desarrollo de una aplicación web demostrativa para la gestión de datos mediante un tablero de cotizaciones de tiempo real con jWebSocket que garantice mayor nivel de usabilidad del marco de trabajo en el desarrollo de aplicaciones financieras en la Web.
4. Validación de la aplicación web desarrollada para garantizar mayor nivel de usabilidad del marco de trabajo jWebSocket en el desarrollo de aplicaciones financieras en la Web.

Los Métodos de la Investigación utilizados fueron:

Métodos Teóricos:

- ✓ **Histórico-Lógico**: Permite analizar la trayectoria completa acerca de la gestión de datos mediante un tablero de cotizaciones en tiempo real en la Web, así como el estudio histórico de las mismas que permite ver deficiencias y proponer soluciones acorde a las necesidades.
- ✓ **Analítico-Sintético**: Mediante este método se analiza toda la fundamentación teórica-metodológica de la investigación recopilada a través de los diferentes medios bibliográficos que pueda servir para desarrollar mejor el diseño de la aplicación web demostrativa, y poder aplicar así estos conocimientos en la práctica de manera que se adquiera una mayor preparación sobre el tema en cuestión.
- ✓ **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.

Método Particular:

- ✓ **Certificación de calidad de software**: Este método se utiliza en la liberación que realiza el equipo de calidad de software del Centro de Desarrollo de la Facultad Regional “Mártires de Artemisa”. Mediante este método se certifica la capacidad funcional de la aplicación web demostrativa y la calidad documental de los artefactos generados durante el proceso de desarrollo.

Se cuentan con las siguientes variables:

- ✓ **Variable Independiente:** Aplicación web para la gestión de datos mediante un tablero de cotizaciones de tiempo real usando el marco de trabajo jWebSocket.
- ✓ **Variable Dependiente:** Nivel de usabilidad del marco de trabajo jWebSocket en el desarrollo de aplicaciones financieras en la Web.

Posibles Resultados:

- ✓ Aplicación web para la gestión de datos mediante un tablero de cotizaciones de tiempo real con jWebSocket que garantice mayor nivel de usabilidad del marco de trabajo en el desarrollo de aplicaciones financieras en la Web.

El contenido de esta investigación ha sido separado en tres capítulos para una mejor comprensión. Conforman además el documento las recomendaciones brindadas por el autor de la investigación, las conclusiones generales, referencias bibliográficas y bibliografía utilizada, anexos y glosario de términos. Los capítulos han sido estructurados de la siguiente manera:

Estructura del Documento:

Capítulo 1. Fundamentación Teórica: Se realiza la fundamentación teórico-

metodológica de la investigación. Se expone un estudio del estado del arte sobre gestión de datos mediante un tablero de cotizaciones de tiempo real en la Web en el ámbito internacional 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 de la aplicación y de la etapa de diseño.

Capítulo 3. Implementación y Validación del Sistema: Se describe la etapa de implementación que conlleva a la obtención de la aplicación demostrativa y se elaboran y documentan las pruebas realizadas a la solución propuesta para demostrar el cumplimiento de los requerimientos de la misma. Se presenta la estrategia de validación diseñada que muestre la capacidad de la aplicación web demostrativa para aumentar los niveles de usabilidad del marco de trabajo jWebSocket en el desarrollo de aplicaciones financieras en la Web.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción

El presente capítulo tiene como objetivo, tratar los principales conceptos y aspectos más significativos asociados al problema y que son necesarios en el desarrollo de la aplicación demostrativa. Se proporciona una panorámica general acerca de la gestión de datos mediante un tablero de cotizaciones en la Web. También se abordará acerca de la metodología, lenguajes de programación, tecnologías y herramientas que serán usados para el desarrollo de la solución.

1.1 Fundamentos Teóricos

En el presente epígrafe se establecen los fundamentos teórico-metodológicos asociados a la gestión de datos a través de tableros de cotizaciones de tiempo real en la Web. Para ello se definen los conceptos de tablero de cotizaciones y tiempo real enmarcados en el objeto de estudio de la investigación. Quedan sentados además los fundamentos metodológicos que guían el proceso de gestión de datos financieros usando tableros de cotizaciones y las características de los tableros de cotizaciones de tiempo real en la Web.

Los tableros de cotizaciones son aplicaciones que muestran el estado de una cotización como principal indicador de las bolsas de valores y de las transacciones de compra y ventas que se realizan entre empresas o personas naturales. El término cotización ha sido establecido por varios autores vinculados al mundo de la gestión financiera. Se encuentra así la definición que plantea que una cotización expresa el rendimiento del mercado accionario en función de las variaciones de precios de una muestra balanceada, ponderada y representativa del conjunto de acciones de la bolsa. Esta definición brinda un grupo de elementos importantes en la conceptualización de cotización sin embargo omite la vinculación directa de las cotizaciones con la compra y ventas de acciones. (Gijón, 2010)

El concepto brindado por Erika D. Báez plantea que la cotización es la tasación oficial que se efectúa de su valor de manera diaria en función de criterios

establecidos anteriormente que depende de las órdenes de compra y venta realizadas. De esta forma queda evidenciada la relación directa entre las cotizaciones y el valor periódico de la compra y ventas de acciones en la gestión financiera. (Erika Dolores Baez Montero, 2009).

Con los elementos tratados anteriormente se define para la investigación que un tablero de cotizaciones es una aplicación que permite que las empresas u otra organización financiera gestione sus cotizaciones. Mediante las cotizaciones las empresas puede introducir órdenes de compra y venta de acciones u otros productos del mercado. Todos estos datos varían de manera periódica y dinámica en el tiempo.

La variación de la información de manera constante en cortos períodos de tiempo, introduce la necesidad de lograr una actualización inmediata de los mismos. Para garantizar este efecto se hace necesario que la aplicación informática que soporta el proceso de gestión de los datos logre tiempo real en su comunicación con el servidor que brinda la información. Tiempo real en la Web ha sido un concepto dado desde distintos enfoques. Muchos autores plantean que una aplicación web garantiza tiempo real cuando el rendimiento de la misma alcanza valores mínimos. (Surhone, y otros, 2010)

Para la presente investigación se toma el concepto de tiempo real en la Web dado por Alexander Schulze, líder de la comunidad jWebSocket. Este plantea que tiempo real en la Web significa lograr dos elementos: el cliente recibe mensajes del servidor sin solicitud previa y los usuarios finales reciben actualizaciones de forma simultánea. (Framework Approach for WebSockets, 2011). De esta manera queda establecida una definición aportada desde la perspectiva del arquitecto principal del marco de trabajo que contituye el centro de la presente investigación.

Los datos que se gestionan mediante un tablero de cotizaciones constituyen en gran medida información numérica y financiera. Este proceso debe cumplir con requisitos que muestren de manera eficiente y con altos niveles de seguridad las

operaciones realizadas por este tipo de aplicaciones. Entre las principales operaciones se encuentran la transacción de acciones, la representación de las fuentes de financiamiento de las empresas y el acceso a información que promuevan que las operaciones financieras se realicen de manera libre, eficiente, competitiva, equitativa y transparente, respetando en todo momento las reglas que rigen la operación del mercado.(Fernández, 2008).

Todos estos requisitos necesarios en la gestión de datos mediante un tablero de cotizaciones han estado presentes en el desarrollo de la presente investigación. De esta manera se garantiza que el resultado del presente trabajo cumpla con los paradigmas establecidos de la gestión financiera en la Web y brinde a los tableros realizados con el marco de trabajo jWebSocket la eficiencia, seguridad y fiabilidad necesarios para su correcto funcionamiento.

1.2 Estado del Arte de soluciones existentes

El estado del arte de la presente investigación se enmarca en el análisis de aplicaciones web de tableros de cotizaciones. Se analizan tableros de cotizaciones en la Web que logran tiempo real a través de las técnicas HTTP *polling* y *long polling*. Se analiza además las aplicaciones de tableros de cotizaciones que logran una comunicación en tiempo real mediante el protocolo WebSocket.

Entre las principales aplicaciones web de tableros de cotizaciones se encuentra *Stock Ticker Application Bar*. Esta es un software con licencia comercial diseñado para la recuperación continua en tiempo real de la cotización de acciones a través de Internet. La aplicación se visualiza al usuario a través de una barra situada en la parte superior o inferior de la pantalla. La barra muestra una lista de acciones predefinida por el usuario y un índice de símbolos que representan cada una de las empresas de interés. La información se actualiza automáticamente o de manera manual y permite además posicionamiento del usuario en una acción específica, selección de fuentes, alarmas visuales, entre otras herramientas.

Esta aplicación es basada en el protocolo de comunicación HTTP y logra tiempo

real en la gestión de la información a través del uso de la tecnología AJAX con las técnicas de *polling* y *long polling.* Sin embargo el protocolo HTTP es basado en un esquema solicitud-respuesta que establece que siempre debe realizarse una solicitud previa para recibir información actualizada desde el servidor. Este hecho provoca que el tiempo real mediante AJAX *polling* o AJAX *long-polling* no garantice realmente una actualización inmediata de los datos al cliente en el momento que se genera un nuevo evento en el servidor.

Otra de las aplicaciones analizadas de tableros de cotizaciones en la Web es *SideTick.* Esta es una aplicación que posee licencia comercial y tiene una manera simple y fácil de mantener el registro de las acciones favoritas de los usuarios. Implementa características importantes entre las que se destaca un rastreador de acciones para ayudar a identificar los valores de las ganancias y las pérdidas. Contiene además un teletipo de acciones que le permite al usuario ver las actualizaciones de los precios de sus acciones favoritas, una función de búsqueda de acciones y variada documentación en línea para lograr mayor usabilidad de la herramienta. Al igual que la herramienta anterior es basada en el protocolo HTTP ocasionando demoras en la entrega de datos al cliente y un aumento del consumo del ancho de banda.

Con el surgimiento y auge de la web móvil y los teléfonos móviles inteligentes han sido creadas aplicaciones web de tableros de cotizaciones que pueden ser accedidas desde esta tecnología. Entre estas aplicaciones se encuentran *StockWatcher,* una aplicación gratuita que permite consultar diariamente el estado de las acciones en el mercado desde un dispositivo móvil con Android. Esta aplicación se sincroniza además con servidores de Yahoo y brinda informes diarios de la bolsa de valores y sus resultados. *StockWatcher* posee además una función de búsqueda de Twitter utilizada que brindan las últimas actualizaciones de las acciones del usuario desde una perspectiva social. Esta aplicación Android utiliza para la comunicación web el protocolo HTTP acarreando todos los problemas que trae consigo esto.

Con el surgimiento del protocolo WebSocket se crean marcos de trabajo para desarrollar aplicaciones web de tableros de cotizaciones en tiempo real. Algunos de los principales servidores que soportan este protocolo son la pasarela Websockets de Kaazing[6], Jetty WebSocket Servlet[7], Socket.IO[8], django-websocket del proyecto Python[9] y jWebsocket.

La pasarela de Websocket de Kaazing es una de las que garantiza hoy tableros de cotizaciones en tiempo real en la Web. En su página oficial posee una aplicación demostrativa que simula el cambio de los valores de las acciones de 3 empresas y muestra una tabla con la tendencia del incremento y decremento de las acciones en tiempo real. La aplicación muestra también el cambio monetario entre dólar y euro donde se observa el cambio constante de los valores en tiempo de ejecución. Sin embargo Kaazing es hoy una pasarela que brinda servicios a través de una licencia comercial y no constituye un marco de trabajo para el desarrollo de aplicaciones con el protocolo WebSocket.

jWebSocket por su parte es un marco de trabajo con un núcleo estable en el lenguaje de programación Java que permite la construcción de aplicaciones estacionarias y móviles usando el protocolo WebSocket. Este marco posee un conjunto de potencialidades para desarrollar tableros de cotizaciones en tiempo real en la Web. Sin embargo no posee una vía que muestre estas potencialidades de manera directa a la comunidad de desarrolladores de aplicaciones financieras. La presente investigación luego del análisis de las soluciones existentes va en la búsqueda de brindar al marco de trabajo jWebSocket la posibilidad de mostrar su capacidad para el desarrollo de tableros de cotizaciones en tiempo real.

---

[6]http://kaazing.com/

[7]http://www.eclipse.org/jetty/

[8]http://socket.io/son

[9]http://pypi.python.org/pypi/django-websocket

1.3 Metodología a utilizar

Todo proceso de desarrollo de una aplicación informática debe estar regido y orientado por una metodología de desarrollo de software, que guíe los procesos y permita tener un registro detallado del avance de la investigación. Las metodologías pueden ser robustas o ágiles. Las metodologías robustas o pesadas están concebidas para guiar el proceso de desarrollo de los software de gran envergadura, cuando un proyecto requiere de gran cantidad de documentación, vaya a ser realizado en un tiempo considerablemente largo y existe la posibilidad de que pase por las manos de varios equipos de trabajo.

Por su parte las metodologías ágiles intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en los clientes y los resultados. Se basan en promover iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto, logrando que se minimicen los riesgos desarrollando software en cortos tiempo. Para el desarrollo de la aplicación web para la gestión de datos mediante un tablero de cotizaciones de tiempo real con jWebSocket se valoraron un grupo de metodologías, de las cuales se muestran sus principales características a continuación.

1.3.1 Proceso Unificado de Desarrollo - RUP

RUP es un proceso formal que provee un acercamiento disciplinado para asignar tarea y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales, respetando cronograma y presupuesto. Fue desarrollado por Rational Software y está integrado con toda la suite de herramientas Rational. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Es guiado por casos de uso y centrado en la arquitectura, iterativo e incremental y utiliza UML como lenguaje de notación. (Figueroa, y otros, 2011)

Consta de 4 fases principales:

- ✓ **Inicio:** el objetivo en esta etapa es determinar la visión del proyecto.

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

1.3.2 SCRUM

Es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software. El desarrollo se realiza de forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nuevas funcionalidades. SCRUM se enfoca en priorizar el trabajo en función del valor que tenga para el negocio, maximizando la utilidad de lo que se construye y el retorno de inversión. (Schwaber, y otros, 2011)

1.3.3 XP

XP es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. Esta consiste en una programación rápida o extrema y es utilizada para proyectos de corto plazo, con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. (Wells, 2009)

1.3.4 SXP

SXP está compuesta por las metodologías SCRUM y XP, ofreciendo una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva. Esta metodología fomenta el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo.

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 donde se realiza la implementación del sistema hasta que esté listo para ser entregado;
- ✓ **Entrega:** es la puesta en marcha; y por último.
- ✓ **Mantenimiento:** es la fase donde se realiza el soporte para el cliente.

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

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

1.3.5 Fundamentos de la selección

Para el desarrollo de la aplicación web que es objetivo de la presente investigación fue seleccionada la metodología SXP. Esta selección es basada en que SXP es una metodología ágil que une las bondades de las metodologías XP y SCRUM, tomando de XP todo el proceso de ingeniería que establece y de SCRUM los elementos de gestión de proyecto y el trabajo en equipo. Por otra parte el proyecto que guía esta investigación está enmarcado en el Centro de Desarrollo de la Facultad Regional “Mártires de Artemisa” el cual cuenta con bibliografía publicada de la metodología.

El Centro tiene además profesores que han utilizado la metodología en proyectos

anteriores y poseen experiencia. Este factor garantiza una mejor gestión del conocimiento y brinda posibilidades de consultar a especialistas de la temática.

1.4 Lenguajes de programación y modelado.

1.4.1 Lenguaje Modelado Unificado

Lenguaje Modelado Unificado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir la estructura del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. Se decide utilizar UML como lenguaje de modelado en su versión 2.0 ya que permite modelar los artefactos establecidos por la metodología SXP seleccionada para la investigación.

1.4.2 Java

Java e un lenguaje moderno, de alto nivel, que recoge los elementos de programación que típicamente se encuentran en todos los lenguajes de programación, permitiendo la realización de programas profesionales. La tecnología Java está compuesta básicamente por 2 elementos: el lenguaje Java y su plataforma. Con plataforma se refiere a la máquina virtual de Java. Actualmente este lenguaje es el más utilizado para el desarrollo de aplicaciones por las ventajas que brindan sus características. Para el desarrollo de la aplicación web del tablero de cotizaciones de tiempo real es utilizado el lenguaje Java, ya que el marco de trabajo jWebSocket desarrolla en este lenguaje en el lado del servidor.

1.4.3 JavaScript

JavaScript es un lenguaje utilizado para realizar acciones dentro del ámbito de una página web. Aunque no es un lenguaje orientado a objetos se pueden implementar muchas de las características de este paradigma y aplicar diversos patrones de código y diseño permitiendo crear aplicaciones web con un alto grado de calidad. Existen un conjunto de librerías JavaScript que facilitan la implementación de aplicaciones con las más diversas funcionalidades.

Entre las características admirables de este lenguaje se destaca su compatibilidad con la mayoría de los navegadores existentes. En cuanto a sus limitaciones, JavaScript fue diseñado de forma que se ejecutara en un entorno muy limitado que permitiera a los usuarios confiar en la ejecución de los scripts. De esta forma, los scripts de JavaScript no pueden comunicarse con recursos que no pertenezcan al mismo dominio desde el que se descargó el script. Este lenguaje es usado para implementar la parte del cliente de la aplicación web de la investigación ya que es el lenguaje que usa el marco de trabajo jWebSocket.

1.4.4 HTML

El Lenguaje para Marcado de Hipertexto HTML es un lenguaje estático para el desarrollo de sitios web permitiendo estructurar las páginas web, fue desarrollado por el World Wide Web Consortium (W3C) y se ha convertido en un estándar reconocido mundialmente constituyendo el idioma común para todos los navegadores. Los archivos pueden tener las extensiones htm y html.

1.5 Tecnologías usadas para el desarrollo de la solución.

1.5.1 Marcos de trabajo

Un marco de trabajo es un esquema, esqueleto o patrón para el desarrollo y/o la implementación de una aplicación. (¿Qué es un 'framework'?, 2006) En otras palabras, un marco de trabajo se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta, puede incluir soporte de programas, bibliotecas y

un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Los marcos de trabajo son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel para proveer un sistema funcional.
Existen multitud de marcos de trabajo orientados a diferentes lenguajes, funcionalidades, entre otros. Aunque la elección de uno de ellos constituye hoy una selección difícil. Para la presenta investigación se valoraron un grupo de ellos seleccionando finalmente el marco de trabajo jWebSocket, el cual forma parte del campo de acción de la presente investigación.

1.5.2 Marco de Trabajo del lado del Servidor

1.5.2.1 Marco de Trabajo jWebSocket

jWebsocket es un marco de trabajo de código abierto para el desarrollo de aplicaciones web estacionarias y móviles basado en Java en el lado del servidor y en JavaScript del lado del cliente. jWebsocket establece un modelo de *token*. Los tokens son datos abstractos que a través de una estructura jerárquica y una API proporcionan métodos de acceso a los contenidos. Con el objetivo de realizar una abstracción en la manipulación de los diferentes formatos, el marco de trabajo convierte los paquetes de datos entrantes y salientes en *tokens*. El cliente nativo soporta el intercambio de paquetes en los formatos JSON, XML y CSV, que en entornos específicos se pueden utilizar sin la necesidad de manejarlos a través de *tokens*. El cliente jWebsocket tiene una arquitectura de *plug-in* que permite aumentar con facilidad sus funcionalidades.

El servidor jWebsocket está diseñado para funcionar como servidor de comunicaciones o como servidor web, brindando total flexibilidad. En la primera opción jWebsocket proporciona un archivo *.jar*, ofreciendo la ventaja de ejecutarse fácilmente desde una línea de comandos e integrarse a la biblioteca de una aplicación existente de Java. En la actualidad hay algunos servidores que ya soportan Websockets y otros que no, por lo que jWebsocket se integra a servidores

como Tomcat o Apache para lograr una comunicación Websockets. En caso de que los servidores soporten de manera nativa Websockets, como el caso de Jetty o GlassFish, se incluyen las funciones de comunicación del marco de trabajo jWebsocket, pero los motores internos se apagan y el anfitrión se utiliza. Esto asegura que no haya mecanismos de seguridad adicionales.

jWebsocket como servidor web proporciona un conjunto importantes funcionalidades y su arquitectura extensible mediante *plug-in* permite añadir fácilmente características adicionales a un sistema independiente. Por otra parte los administradores pueden configurar el servidor exactamente como sea necesario y dejar a un lado todos los módulos que no necesiten. En un clúster los *plug-ins* se pueden utilizar como servicios, por lo que jWebsocket perfectamente es compatible con Service Oriented Architectures (SOA) en un entorno totalmente basado en eventos. Estas características muestran la fortaleza y flexibilidad del marco de trabajo para el desarrollo de aplicaciones web estacionarias y móviles, multiplataforma, multisectorial y compatible con todos los navegadores. (Framework Approach for WebSockets, 2011)

1.5.3 Fundamentos de la selección.

jWebSocket es un proyecto del cual un equipo de estudiantes de la Facultad Regional de la UCI “Mártires de Artemisa”, constituye alrededor de un 50% de los integrantes del grupo de desarrollo de este marco de trabajo. La presente investigación es parte del proyecto jWebSocket, precisamente por esta razón la aplicación a desarrollar se lleva a cabo sobre este marco de trabajo.

1.5.4 Marcos de Trabajo del lado del Cliente

1.5.4.1 Marco de Trabajo Ext JS 4

Ext JS constituye un gran avance a los marcos de trabajo JavaScript. Con mayor funcionalidad, *plug-in* gratuito para gráficos y una nueva arquitectura Modelo-Vista-Controlador, permite crear aplicaciones web robustas para todos los navegadores. Ext JS 4 trae un paquete de datos completamente nuevo que permite a los

desarrolladores utilizar una arquitectura flexible en la construcción de las aplicaciones en el lado de cliente web. Al permitir la separación de la gestión de datos, la lógica y elementos de la interfaz, Ext JS 4 hace que sea fácil, incluso para los grandes equipos de desarrollo, trabajar de manera independiente sin preocupaciones.

Sobre la base de Ext JS 3.3, la última versión añade más de 350 nuevas API, 50 nuevas clases, y un 65% más de documentación. Un paquete de datos completamente nuevo anima a los desarrolladores a aprovechar funciones como el desplazamiento de cuadrícula infinita para construir un nuevo nivel de interactividad en las aplicaciones web.

Ext JS 4 permite a los desarrolladores entregar sobre una increíble variedad de navegadores y sistemas operativos con el mismo código. En los navegadores modernos utiliza las características de HTML5. Hace que sea fácil crear una aplicación poderosa en la web, independientemente del navegador que el cliente usa.

Ext JS 4 proporciona los gráficos más avanzados y capacidades gráficas de cualquier marco de JavaScript, sin depender de *plug-ins*, ofreciendo una visualización perfecta de píxeles en cualquier navegador sobre cualquier sistema operativo. Aprovechando SVG y VML, Ext JS 4 con su paquete de gráficos, permite a los desarrolladores diseñar y programar sus gráficos en cualquier navegador. (Sencha, 2011)

Introduce una manera muy sencilla de cargar clases bajo demanda, esto permite optimizar la carga inicial de la aplicación e ir cargando los módulos justo cuando el usuario los solicita. (Carga de clases dinámicamente con ExtJS 4, 2011) Entre sus principales **ventajas** se encuentran:

- ✓ La orientación a objetos intensa permite modular todos los scripts.
- ✓ El diseño está completamente separado de la funcionalidad.
- ✓ Funciones comunes como validación, comboxes editables, ventanas

arrastrables (con minimizar y maximizar), grillas editables, son muy fáciles de implementar.

- ✓ Buena y amplia documentación.

1.5.5 Fundamentos de la Selección

Para el desarrollo de este tipo de aplicación existen diversos marcos de trabajo de JavaScript, destacándose jQuery, Moo Tools, Prototype y ExtJS 4 utilizando este último por las múltiples ventajas y funcionalidades que proporciona, facilitando el trabajo con datos en las tablas, ya que posee componentes apropiados específicamente para esto, además la vinculación con el marco de trabajo jWebSocket es de forma sencilla, facilitando en gran medida el desarrollo de la aplicación.

1.6 Herramientas a emplear para el desarrollo de la solución

1.6.1 Herramienta CASE

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

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

1.6.1.1 Visual Paradigm

Visual Paradigm es una poderosa herramienta CASE que hace uso del UML con soporte multiplataforma, que proporciona un ciclo de vida completo del desarrollo

de software, excelentes facilidades de interoperabilidad con otras aplicaciones, compatibilidad entre versiones, así como dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Dicha herramienta brinda una serie de facilidades que se mencionan a continuación:

- ✓ Soporta un conjunto de estándares entre los que se encuentran UML, lenguaje de modelado usa en la investigación.
- ✓ Integración con herramientas Java como Eclipse/IBM WebSphere ,JBuilder , BEA Weblogic, Oracle JDeveloper y NetBeans IDE, esta última utilizada para el desarrollo de la aplicación web del tablero de cotizaciones en tiempo real.
- ✓ Permite la generación de código y la ingeniería inversa para un conjunto de lenguajes entre los que se encuentran Java. Cualquiera de los cambios en el código existente puede reflejarse en el modelo y viceversa.
- ✓ Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.
- ✓ Interoperabilidad e integración. Permite la integración con un conjunto de herramientas e intercambiar diagramas UML y modelos usando representaciones industriales comunes.
- ✓ Modelado de base de datos. Proporciona una mayor documentación de la base de datos y diagramas de mapeo de relación de objetos.
- ✓ Integración con herramientas para el control de versiones como el Subversion.

1.6.1.2 Fundamentos de la selección

Anteriormente existen herramientas para el modelado de software, entre ellas Rational Rose y Visual Paradigm, teniendo en cuenta que se desea desarrollar un trabajo bajo las políticas de estándares de código abierto, se selecciona como herramienta CASE a Visual Paradigm. Esta herramienta a pesar de no ser libre, cuenta con una licencia comercial la cual posee la Universidad de las Ciencias

Informáticas, centro rector del presente Trabajo de Diploma.

1.6.1.3 Herramientas de Control de Versiones

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

Un sistema de control de versiones es un sistema de gestión de archivos y directorios, cuya principal característica es mantener el historial de cambios y modificaciones que se han realizado sobre dichos archivos a lo largo del tiempo. Es importante decir que estos sistemas no solo se limitan a gestionar archivos de texto sino que también gestionan documentos, imágenes y ficheros de todo tipo.

1.6.1.4 Subversion

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

Este sistema presenta varias características importantes. Los directorios son versionados. La resolución de conflictos es de forma interactiva. Gestiona de manera eficaz los archivos binarios. Bloquea los archivos. Vincula los lenguajes de programación. Vincula varios repositorios. Posee soporte para desarrolladores y desarrollo paralelo.

1.6.1.5 Fundamentos de la selección

Existen otras herramientas de control de versiones, como es el caso de Git, pero teniendo en cuenta las características expuestas, se selecciona Subversion como herramientas de control de versiones para ser utilizado en el desarrollo de la aplicación, ya que permite la integración con NetBeans y la posibilidad de desarrollar en paralelo. También influyó en la decisión, la experiencia de trabajo por

parte del equipo de desarrollo con esta herramienta y su fácil uso.

1.6.1.6 Herramientas del Cliente de Control de Versiones

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

1.6.1.7 RapidSVN

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

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

Características:

- ✓ Simple: Proporciona una interfaz fácil de usar para las características de Subversion.
- ✓ Eficiente: Simple para los principiantes pero lo suficientemente flexible como para aumentar la productividad para los usuarios de Subversion.
- ✓ Portátil: Se ejecuta en cualquier plataforma: Linux, Windows, Mac OS / X, Solaris.
- ✓ Rápido: Completamente escrito en C + +.

1.6.1.8 Fundamentos de la selección

Actualmente existen herramientas de control de versiones del lado del cliente, como por ejemplo Tortoise, a pesar de todas las ventajas y funcionalidades que ofrece Tortoise, se decide usar Rapidsvn ya que ésta es libre, y fácil de usar, tanto por quienes ya conocen Subversion como para quienes empiezan, pudiendo acceder a direcciones SVN, subir y descargar contenido, sincronizarlo con el

servidor original, comprobar su estado, crear y fusionar direcciones.

1.6.2 Entorno integrado de desarrollo - IDE

Un IDE (acrónimo en inglés de Integrated Development Environment) es un entorno de programación que integra varias herramientas con el objetivo de facilitar el desarrollo de software sobre uno o varios lenguajes de programación. La mayoría de los IDEs cuentan con herramientas tales como: editor de código, herramientas para el rastreo de código, compilador, depurador y constructor de interfaz gráfica (Nourie, 2005).A continuación se hace una caracterización de varios IDEs que permitirá adquirir los elementos necesarios para determinar cuál es el más idóneo para el desarrollo de solución propuesta.

1.6.2.1 NetBeans

NetBeans IDE es una herramienta desarrollada por Sun Microsystems. Está completamente escrito en Java, por lo que puede ser utilizado desde cualquier sistema operativo compatible con la máquina virtual de Java. Permite el desarrollo de aplicaciones de escritorio, web y móviles. Brinda soporte a varios lenguajes de programación como Java, PHP, C/C++, Groovy, Python, JavaScript, entre otros. Es un producto libre y gratuito sin restricciones de uso. Este entorno integrado de desarrollo es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Su misión consiste en evitar tareas repetitivas, facilitar la escritura correcta de código, disminuir el tiempo de depuración e incrementar la productividad del desarrollador. Cuenta con un depurador, perfilador de integración, herramientas para refactorizaciones, completamiento de código y control de versiones de archivos. (NetBeans, 2011).

1.6.2.2 Fundamentos de la solución

Las características anteriormente expuestas sobre NetBeans IDE y Eclipse SDK demuestran las potencialidades de ambos para el desarrollo de aplicaciones Java. Sin embargo, para el desarrollo de la solución propuesta se selecciona como entorno integrado de desarrollo a NetBeans IDE debido a que se tiene una mayor

experiencia y familiarización con esta herramienta. Además su versión 7.0.1 introduce un soporte para el desarrollo con la especificación JavaSE7(Java Standard Edition) con las característicasdeJDK7(Java Development Kit).Esta versión también ofrece una integración mejorada con el servidor OracleWeb Logic, así como soporte para Oracle Database y GlassFish3.1.Otros puntos destacados incluyen soporte para Maven3 yHTML5, así como mejoras en el editor de Java (NetBeans, 2011).

1.6.2.3 Sistema Gestor de Base de Datos

Los sistemas de gestión de bases de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Su propósito es manejar de forma clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización.

Las ventajas que proporcionan los gestores de base de datos son:

- ✓ Proveen facilidades para la manipulación de grandes volúmenes de datos.
- ✓ Simplifican la programación de equipos de consistencia.
- ✓ Manejando las políticas de respaldo adecuadas, garantizan que los cambios de la base serán siempre consistentes sin importar si hay errores correctamente, etc.
- ✓ Organizan los datos con un impacto mínimo en el código de los programas.
- ✓ Disminuyen drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado si son bien explotados por los desarrolladores.
- ✓ Usualmente, proveen interfaces y lenguajes de consulta que simplifican la recuperación de los datos.

Existen bases de datos relacionales y documentales (o no relacionales). El auge de la Web está mirando las bases de datos documentales como clave en sus negocios, por su mayor capacidad de escalabilidad, rendimiento y alta

disponibilidad que los actuales sistemas de bases de datos relacionales.

En las bases de datos documentales los datos están fuertemente orientados a documentos, haciéndolos más acordes y naturales con el modelo de datos clave/valor que con el modelo de datos relacional. El entorno de desarrollo está fuertemente orientado a objetos, y una base de datos clave/valor podría reducir la necesidad de código tubería. El almacenamiento de datos es económico y se integra fácilmente con la plataforma de servicios web de tu proveedor.

Características de las bases de datos documentales

Las bases de datos documentales a diferencia de las bases de datos relacionales, están libres de esquemas y de estructuras. Por tanto, la información no se divide en columnas fijas y filas, sino en documentos que pueden tener libertad de información, como cualquier cantidad de campos o de tipos de campos, e incluso de contener en el propio campo una lista de valores, un documento o una colección de documentos.

A continuación se especifican las principales características de las bases de datos relacionales tenidas en cuenta para esta investigación.

- ✓ La información almacenada en un documento está formada por pares de clave y valor.
- ✓ En cuanto a estructura de datos, las bases de datos documentales son libre de esquemas. Al no estar sujeta a estructuras ni relaciones, los cambios de diseño son fáciles de adoptar y con un impacto mínimo.
- ✓ Son muy sencillas y potencia en escalabilidad.
- ✓ Estas bases de datos son muy eficientes a la hora de explotar grandes volúmenes de información textual.
- ✓ Los sistemas de bases de datos documentales ganan rapidez dedicándose más en exclusiva al almacenamiento y la recuperación de la información.
- ✓ La gran ventaja de no poseer esquema y la consiguiente rigidez de

estructuras y relaciones, es lo que las hace muy eficientes al trabajar con grandes cantidades de documentos.

- ✓ Los campos vacíos no se añaden al documento, optimizando el espacio de almacenamiento.
- ✓ Posibilidad en embeber documentos y colecciones dentro de documentos.
- ✓ Poseen un lenguaje de consulta natural.

Entre los principales gestores de bases de datos documentales se encuentra MongoDB utilizado en el almacenamiento y seguridad de los datos en la presente investigación.

1.6.2.4 MongoDB

Mongo es el sistema de base de datos desarrollada en 10gen por Geir Magnusson y Dwight Merriman. MongoDB es una base de datos orientada a documentos JSON, salvo que está diseñada para ser una verdadera base de datos de objetos, más que para un almacenamiento de clave/valor puro. Originalmente, 10gen enfoca poner juntos una pila completa de servicios web, aunque sin embargo, más recientemente ha tenido que ser reenfocado principalmente en la base de datos MongoDB.

1.6.2.5 Fundamentos de la solución

Existen numerosas bases de datos como son SimpleDB, CouchDB, Cassandra, entre otras. Basado en las características expuestas anteriormente se hace uso de la base de datos documental MongoDB. Esta supera en gran medida a las restantes bases de datos ya que es el enlace perfecto entre el almacenamiento clave/valor (que son rápidos y altamente veloces) y los sistemas tradicionales de RDBMS (que proporcionan consultas ricas y una profunda funcionalidad).

MongoDB es escalable, de alto desempeño, de código abierto, base de datos orientada a documentos. Escrito en C++ ofrece las siguientes características:

- ✓ Almacenamiento orientado a documentos: Documentos estilo JSON con

esquemas dinámicos ofrecen simplicidad y poder.

- ✓ Soporte Full índex: Índices sobre cualquier atributo, tal y como estamos acostumbrados.
- ✓ Replicación y alta disponibilidad: Espejos entre LANs y WANs
- ✓ Auto-Sharding: Escalabilidad horizontal sin comprometer la funcionalidad.
- ✓ Consultas: Ricas y basadas en documentos
- ✓ Rápidas actualizaciones en el contexto.
- ✓ Mapeo y reducción: Agregación flexible y procesamiento de datos.
- ✓ GridFS: Almacena archivos de cualquier tamaño sin complicar el “stack”.
- ✓ Soporte comercial: Soporte comercial, capacitación y consultoría disponibles.

Conclusiones del capítulo

En el presente capítulo fueron tratados los principales conceptos y aspectos más significativos asociados a la gestión de tableros de cotizaciones en tiempo real en l Web. Se analizaron un conjunto de aplicaciones que implementan hoy tableros de cotizaciones en la Web tanto con el uso del protocolo HTTP como con WebSocket. Son seleccionadas la metodología, lenguajes de programación, tecnologías y herramientas que son usados para el desarrollo de la solución.

CAPÍTULO 2. CARACTERÍSTICAS, ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

En el presente capítulo se describen las funcionalidades de la aplicación web, destacando sus características principales. Los artefactos que posee la metodología de desarrollo SXP, propuesta en el capítulo anterior, son brevemente detallados y explicados. Para lograr entender mejor su contexto se define el modelo de dominio y se describen los requisitos funcionales y no funcionales de la aplicación web, las historias de usuario y las tareas de ingeniería asociadas a las mismas.

2.1 Propuesta de Solución

El marco de trabajo jWebSocket es capaz de lograr la visualización de datos de los tableros de cotización solucionando las limitaciones que presenta el protocolo HTTP. Para ello se desarrolla en el presente trabajo de diploma la aplicación Stock Ticker Demo que muestre datos en tiempo real de un tablero de cotizaciones utilizando este marco de trabajo.

La aplicación permite a los usuarios ver el comportamiento de las acciones de las empresas en tiempo real, hacer un análisis del precio mediante un historial, hacer su compra-venta de acciones. Estas actividades del usuario son reflejadas en una tabla de depósitos, donde se puede ver el comportamiento de cada depósito mediante una gráfica donde se refleja el valor que va tomando el dinero con respecto al tiempo, de una empresa seleccionada.

Stock Ticker Demo brinda a los usuarios numerosos beneficios tales como, hacer un análisis del valor de las finanzas para tomar decisiones de inversión sobre las mismas, disminuyendo los riesgos a tener pérdidas de sumas monetarias. Además mediante una gráfica de su depósito de compras se puede obtener de manera más eficiente el comportamiento de las inversiones del usuario con respecto al tiempo.

Stock Ticker Demo es realizada mediante el protocolo de comunicación WebSocket, el cual proporciona que se logre mostrar datos en tiempo real a altos

niveles de velocidad y escalabilidad. jWebSocket es el marco de trabajo en el que fue desarrollada la aplicación Stock Ticker Demo, mostrando las potencialidades que éste ofrece para mostrar datos de los tableros de cotizaciones en la Web en tiempo real.

A continuación en la **Fig. 1** se muestra la propuesta de solución realizada en la presente investigación.

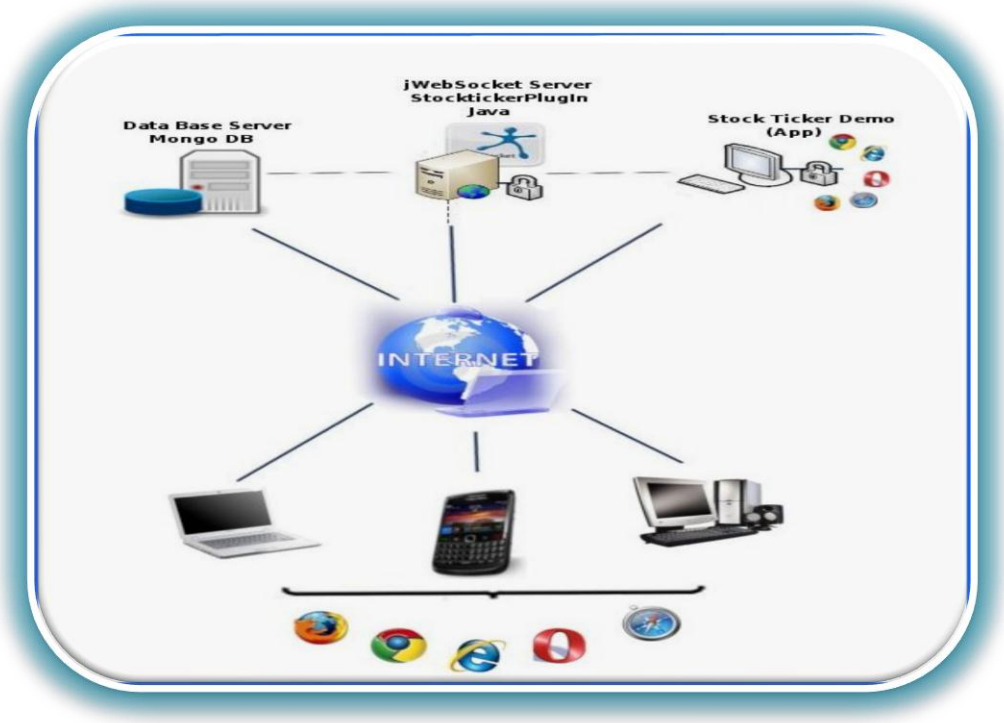


Fig. 1: Propuesta de solución. Fuente: Elaboración propia.

2.2 Planificación del Proyecto por Roles

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	Yamila Vigil Regalado

	del producto. Coordina y facilita las reuniones. Asegura que se consiguen los objetivos de cada iteración.	
Gerente	Es el responsable de tomar las decisiones finales, acerca de estándares y convenciones a seguir durante el proyecto. Participa en la selección de objetivos y requerimientos. Tiene la responsabilidad de controlar el progreso y trabaja junto con el Jefe de Proyecto en la reducción de la Lista de Reserva del Producto.	Alexander Schulze
Especialista	Es necesario que conozca a fondo el proceso para el desarrollo de software. Es una especialización que está activa, el miembro del grupo de trabajo que la desempeña siempre está ejecutándola y alcanzando un grado mayor de conocimientos en el tema, en este caso como Diseñador Gráfico.	Rebecca Schulze
Consultor	Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas, además aportan ideas y experiencias para el beneficio del sistema en desarrollo. Esta es una especialización menos activa, quien la ejecuta funciona en este rol por un corto período de tiempo.	Alexander Schulze Rebecca Schulze
Cliente	Participa en las tareas que involucran la lista de reserva del producto.	Alexander Schulze
Programador	Elabora el código de las nuevas funcionalidades a implementar. Escribe las pruebas unitarias. Debe existir una comunicación y coordinación adecuada entre los programadores y el resto del equipo.	Roylandi González Pujol
Analista	Escribe las historias de usuario y las pruebas funcionales para validar su implementación.	Roylandi González Pujol
Diseñador	Encargado del diseño del sistema y de los prototipos de interfaces, son los máximos responsables de la realización	Roylandi González Pujol

	del diseño de las metáforas y supervisan el proceso de construcción.	
Encargado de Pruebas	Es el encargado de ayudar al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.	Roylandi González Pujol
Arquitecto	Se vincula con el analista y el diseñador ya que su trabajo tiene que ver con la estructura y el diseño del sistema. Ayuda en el diseño de las metáforas.	Roylandi González Pujol

2.3 Modelo de Historias de Usuario del Negocio

Dentro de los artefactos que genera la metodología SXP se encuentra la plantilla del Modelo Historias de Usuario del Negocio, donde se definen las características específicas del negocio en cuestión.

Este modelo describe además los pilares fundamentales del proceso de gestión financiera en un tablero de cotizaciones. Visualizar los datos en un tablero de cotizaciones en la Web permite que los usuarios analicen la información, además les permite seleccionar las acciones que desea comprar o vender del tablero de cotizaciones. A continuación se muestra en la **Fig. 2** el modelo de negocio correspondiente a la solución.

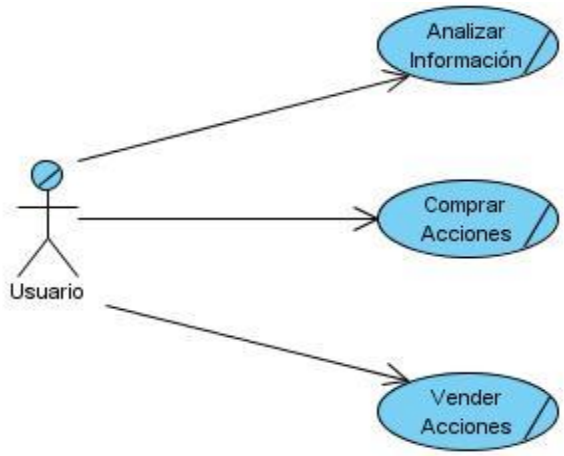


Fig. 2: Modelo de Negocio. Fuente: Elaboración propia.

2.4 Lista de Reserva del Producto (LRP)

La lista de Reserva del Producto es una lista con prioridad en la cual queda reflejada todas las tareas a desarrollar en el proyecto durante el tiempo de desarrollo. A medida que vayan aumentando los conocimientos relacionados al producto y se vaya identificando con más claridad las necesidades del cliente, irá aumentando o variando esta lista, estos cambios solo pueden hacerse entre iteraciones. La presente investigación cuenta con un total de 8 requisitos funcionales y 20 requisitos no funcionales.

A continuación en la **Tabla 2** se muestra la lista de reserva del producto:

Tabla 2: Lista de Reserva del Producto

Prioridad	Ítem *	Descripción	Estimación	Estimado por
Muy Alta				
	1	Mostrar la tabla de cotizaciones.	1	Analista
	2	Mostrar historial.	1	Analista
	3	Comprar instrumentos.	2	Analista
	4	Vender instrumentos.	2	Analista
	5	Graficar depósitos.	1	Analista
	6	Mostrar depósitos.	1	Analista
Alta				
	7	Registrar usuario.	2	Analista
	8	Autenticar usuario	2	Analista
Media				
Baja				
RNF (Requisitos No Funcionales)				
	9	La aplicación está dirigida a usuarios con conocimientos básicos de informática y un correcto dominio del negocio en		

		cuestión, por lo cual debe estar diseñada para ser de fácil manejo por el cliente.		
	10	El Sistema es una aplicación web.		
	11	El mantenimiento de la aplicación se realizará en tiempo de ejecución sin afectar la disponibilidad ni el rendimiento de los servicios.		
	12	Garantizar la integridad y consistencia de los datos.		
	13	Se debe realizar la aplicación de forma versionable que permita darle mantenimientos al sistema a fin de aumentar las funcionalidades y/o corregir los errores del mismo a través de versiones posteriores.		
	14	Se documentará la aplicación con diferentes manuales con el objetivo de explicar el uso de la aplicación para garantizar el soporte de la misma.		
	15	En el lado del servidor el lenguaje de programación a utilizar es Java, empleando el Framewok jWebSocket y el IDE NetBeans 7.0.1.		
	16	En el lado del cliente se hará uso de los lenguajes JavaScript y HTML, empleando el FramewokSencha y el IDE NetBeans 7.0.1.		
	17	La metodología de desarrollo a seguir es SXP y para la modelación se utilizará la herramienta Visual Paradigm 3.4.		
	18	Para el desarrollo de la aplicación en el lado del cliente se utilizará una arquitectura en capas MVC.		
	19	El gestor de BD a utilizar es MongoDB v2.0.		
	20	Lograr un diseño amigable y de		

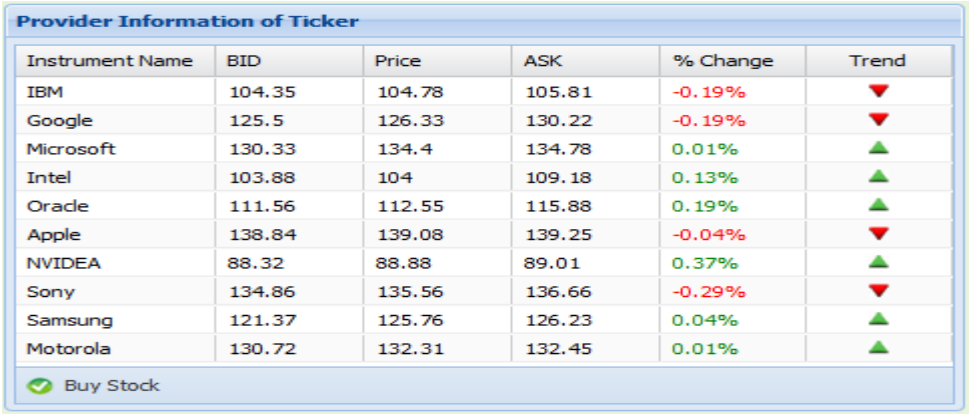
		fácil uso.		
	21	Todas las interfaces de usuario que se definan para el sistema respetarán los patrones de diseño establecidos para la organización.		
	22	Los requisitos mínimos de hardware para el correcto funcionamiento de la aplicación son: 512 MB de memoria RAM, microprocesador Pentium IV, tarjeta red 100 Mbps.		
	23	Los requisitos mínimos de hardware para el servidor de Base de Datos son: 2GB de RAM.		
	24	El cliente debe contar con un navegador que soporte el protocolo WebSocket.		
	25	El servidor de aplicaciones debe tener instalado la máquina virtual de Java OpenJDK 7 y el servidor de jWebSocket.		
	26	El servidor de base de datos requiere un sistema operativo de 64 bits.		
	27	El código de la aplicación será liberado bajo la Licencia Pública General Reducida de GNU (LGPL)		
	28	El código y la documentación de la aplicación deberán cumplir estrictamente las normas de calidad estipuladas por el grupo de CALISOFT de la UCI.		

2.5 Historias de Usuario y Tareas de Ingeniería

Las historias de usuario en la metodología de desarrollo SXP son las que describen las tareas que el sistema debe hacer, cuestión que depende en gran medida de las especificaciones realizadas por el cliente. 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 en la **Tabla 3** se muestran cada una de las historias de usuarios con sus respectivas tareas asociadas:

Tabla 3: Historia de Usuario Mostrar tablero de cotizaciones.

<table>
<tr><th colspan="2">Historia de Usuario</th></tr>
<tr><td>Número:HU_1</td><td>Nombre Historia de Usuario: Mostrar tablero de cotizaciones.</td></tr>
<tr><td colspan="2">Modificación de Historia de Usuario Número: Ninguna</td></tr>
<tr><td>Usuario: Roylandi González Pujol</td><td>Iteración Asignada:2</td></tr>
<tr><td>Prioridad en Negocio: Muy Alta</td><td>Puntos Estimados:1</td></tr>
<tr><td>Riesgo en Desarrollo: Alto</td><td>Puntos Reales: 1</td></tr>
<tr><td colspan="2">Descripción: La presente historia de usuario permitirá al usuario interactuar con la tabla de cotizaciones que es donde se reflejan el estado de las cotizaciones de cada organización teniendo en cuenta el precio de las acciones, la oferta-demanda y la tendencia de los precios en el tiempo.</td></tr>
<tr><td colspan="2">Observaciones: Para que esto sea posible el usuario debe haberse autenticado en la aplicación.</td></tr>
<tr><td colspan="2">Prototipo de interface:
 
</td></tr>
</table>

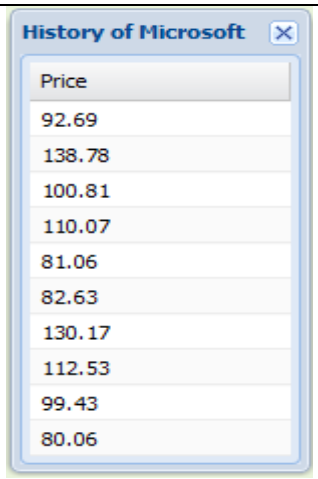
A continuación se muestra en la **Tabla 4** la descripción de la tarea de ingeniería que se realiza para lograr el cumplimiento de la historia de usuario anterior:

Tabla 4: Tarea de Ingeniería Mostrar la tabla de cotizaciones.

Tarea de Ingeniería	
Número Tarea: 1.1	**Número Historia de Usuario:** HU_1
Nombre Tarea: Mostrar la tabla de cotizaciones.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 21/11/2011	**Fecha Fin:** 25/11/2011
Programador Responsable: Roylandi González Pujol	
Descripción: Desarrollar la opción de mostrar la tabla de cotizaciones el cual permitirá al usuario interactuar con la tabla de cotizaciones que es donde se reflejan el estado de las cotizaciones de los instrumentos.	

Tabla 5: Historia de Usuario Mostrar historial.

Historia de Usuario	
Número:HU_2	**Nombre Historia de Usuario:** Mostrar historial.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roylandi González Pujol	**Iteración Asignada:***2*
Prioridad en Negocio: Muy *Alta*	**Puntos Estimados:***1*
Riesgo en Desarrollo: *Alto*	**Puntos Reales:** 1
Descripción: La presente historia de usuario tiene como objetivo permitir que el usuario pueda ver el comportamiento de los últimos 10 precios que tomaron los instrumentos de una empresa específica.	
Observaciones: Para que esto sea posible el usuario debe dar doble click encima del instrumento que quiere ver su historial.	
Prototipo de interface:	

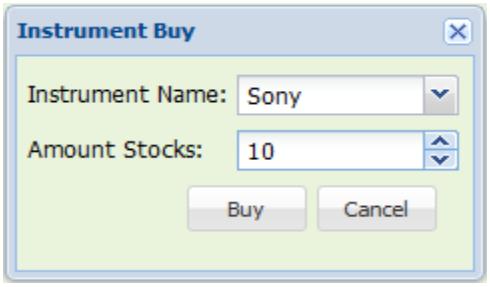


A continuación se muestra en la **Tabla 6**, la descripción de la tarea de ingeniería que se realiza para lograr el cumplimiento de la historia de usuario anterior:

Tabla 6: Tarea de Ingeniería Mostrar historial.

Tarea de Ingeniería	
Número Tarea: 2.1	**Número Historia de Usuario:** HU_2
Nombre Tarea: Mostrar historial.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 29/11/2011	**Fecha Fin:** 2/12/2011
Programador Responsable: Roylandi González Pujol	
Descripción: Desarrollar la opción de mostrar historial la cual permite que el usuario pueda ver el comportamiento de los últimos 10 precios que tomaron los instrumentos de una empresa específica.	

Tabla 7: Historia de Usuario Comprar instrumentos.

Historia de Usuario	
Número:HU_3	**Nombre Historia de Usuario:** Comprar instrumentos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roylandi González Pujol	**Iteración Asignada:**2
Prioridad en Negocio: Muy *Alta*	**Puntos Estimados:**2
Riesgo en Desarrollo: Muy *Alto*	**Puntos Reales:** 2
Descripción: La presente historia de usuario tiene como objetivo permitir que el usuario pueda comprar instrumentos en el tablero de cotizaciones.	
Observaciones: Para que esto sea posible el usuario debe acceder al formulario comprar mediante el botón “Buy Stock” escoger el instrumento que quiere comprar y la cantidad de acciones.	
Prototipo de interface: 	

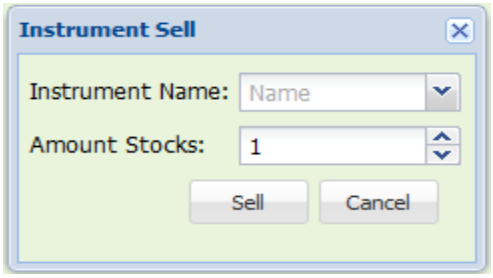
A continuación se muestra en la **Tabla 8**, la descripción de la tarea de ingeniería que se realiza para lograr el cumplimiento de la historia de usuario anterior:

Tabla 8: Tarea de Ingeniería Comprar instrumentos.

Tarea de Ingeniería	
Número Tarea: 3.1	**Número Historia de Usuario:** HU_3
Nombre Tarea: Comprar instrumentos.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2

Fecha Inicio: 5/12/2011	**Fecha Fin:** 14/12/2011
Programador Responsable: Roylandi González Pujol	
Descripción: Desarrollar la opción de comprar instrumentos en la tabla de cotizaciones la cual permite acceder al formulario comprar mediante el botón “Buy Stock” escoger el instrumento que quiere comprar y la cantidad de acciones.	

Tabla 9: Historia de Usuario Vender instrumentos.

Historia de Usuario	
Número:HU_4	**Nombre Historia de Usuario:** Vender instrumentos
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roylandi González Pujol	**Iteración Asignada:**2
Prioridad en Negocio: Muy *Alta*	**Puntos Estimados:**2
Riesgo en Desarrollo: Muy *Alto*	**Puntos Reales:** 2
Descripción: La presente historia de usuario tiene como objetivo permitir que el usuario pueda vender instrumentos de su depósito personal.	
Observaciones: Para que esto sea posible el usuario debe tener compras de instrumento que quiere vender en su depósito personal acceder al formulario vender mediante el botón “Sell Stock” escoger el instrumento que quiere vender y la cantidad que quiere vender.	
Prototipo de interface: 	

A continuación se muestra en la

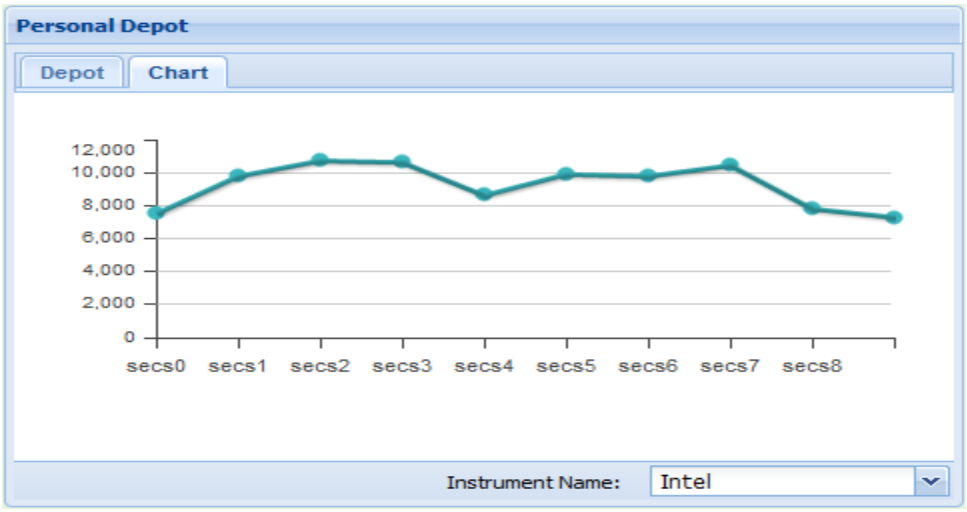
Tabla 10, la descripción de la tarea de ingeniería que se realiza para lograr el cumplimiento de la historia de usuario anterior:

Tabla 10: Tarea de Ingeniería Vender instrumentos.

Tarea de Ingeniería	
Número Tarea: 4.1	**Número Historia de Usuario:** HU_4
Nombre Tarea: Vender instrumentos.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 19/12/2011	**Fecha Fin:** 13/01/2012
Programador Responsable: Roylandi González Pujol	
Descripción: Desarrollar la opción de vender instrumentos de la tabla de compras que permite que el usuario pueda vender instrumentos de su depósito personal.	

Tabla 11: Historia de Usuario Graficar depósitos.

Historia de Usuario	
Número:HU_5	**Nombre Historia de Usuario:** Graficar depósitos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roylandi González Pujol	**Iteración Asignada:**2
Prioridad en Negocio: Muy *Alta*	**Puntos Estimados:**1
Riesgo en Desarrollo: *Alto*	**Puntos Reales:** 1
Descripción: La presente historia de usuario tiene como objetivo permitir que el usuario pueda ver el comportamiento de sus depósitos a intervalos configurables. Se podrá observar como varía el valor de sus depósitos en función del tiempo.	
Observaciones: Para que esto sea posible el usuario debe tener compras en	

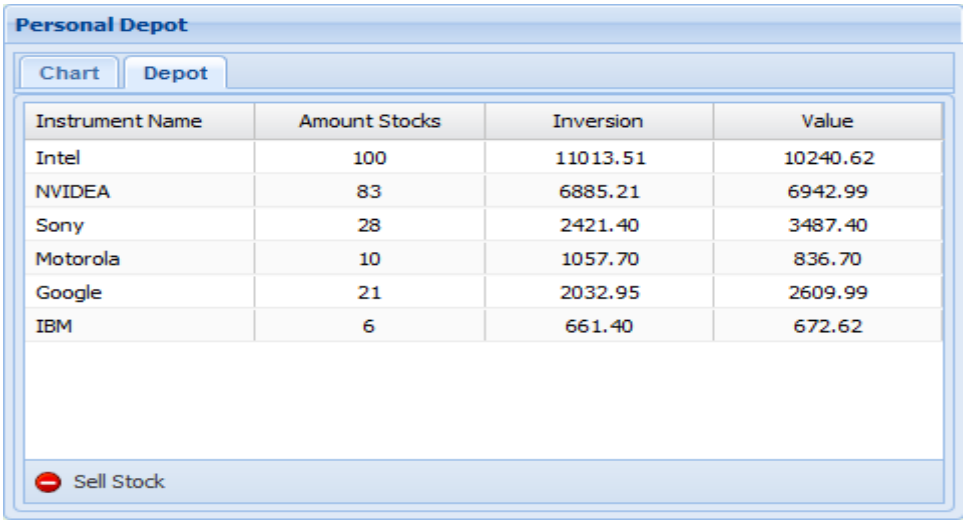
su depósito personal.
Prototipo de interface: 

A continuación se muestra en **Tabla 12**, la descripción de la tarea de ingeniería que se realiza para lograr el cumplimiento de la historia de usuario anterior:

Tabla 12: Tarea de Ingeniería Graficar depósitos.

Tarea de Ingeniería	
Número Tarea: 5.1	**Número Historia de Usuario:** HU_5
Nombre Tarea: Graficar depósitos.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 17/01/2012	**Fecha Fin:** 20/01/2012
Programador Responsable: Roylandi González Pujol	
Descripción: Desarrollar la opción de graficar depósitos que permite que el usuario pueda ver el comportamiento de sus compras a intervalos configurables. Se podrá observar como varía el valor de sus compras en función del tiempo.	

Tabla 13: Historia de Usuario Mostrar depósitos.

Historia de Usuario	
Número:HU_6	**Nombre Historia de Usuario:** Mostrar depósitos.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roylandi González Pujol	**Iteración Asignada:**2
Prioridad en Negocio: Muy *Alta*	**Puntos Estimados:**1
Riesgo en Desarrollo: *Alto*	**Puntos Reales:** 1
Descripción: La presente historia de usuario permitirá al usuario interactuar con la tabla de depósitos que es donde se reflejan el estado de las compras y ventas de los instrumentos.	
Observaciones: Para que esto sea posible el usuario debe haberse autenticado en la aplicación.	
Prototipo de interface: 	

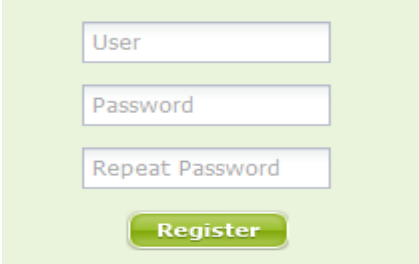
A continuación se muestra en la **Tabla 14** la descripción de la tarea de ingeniería que se realiza para lograr el cumplimiento de la historia de usuario anterior:

Tabla 14: Tarea de Ingeniería Mostrar depósitos.

Tarea de Ingeniería	
Número Tarea: 6.1	**Número Historia de Usuario:** HU_6
Nombre Tarea: Mostrar depósitos.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 23/01/2012	**Fecha Fin:** 27/01/2012
Programador Responsable: Roylandi González Pujol	
Descripción: Desarrollar la opción de mostrar depósitos que usuario permite al usuario interactuar con la tabla de depósitos que es donde se reflejan el estado de las compras y ventas de los instrumentos.	

Tabla 15: Historia de Usuario Registrar usuario.

Historia de Usuario	
Número:HU_7	**Nombre Historia de Usuario:** Registrar usuario.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roylandi González Pujol	**Iteración Asignada:**3
Prioridad en Negocio: ***Alta***	**Puntos Estimados:**2
Riesgo en Desarrollo: Muy *Alto*	**Puntos Reales:** 2
Descripción: La presente historia de usuario tiene como objetivo permitir el registro del usuario en el sistema para que pueda acceder a la aplicación e interactuar con la misma.	
Observaciones: Para que esto sea posible el usuario debe dirigirse al “Demo de Tablero de cotizaciones” que se encuentra en la pestaña “Demos” del sitio de jWebSocket y registrarse en el demo de “Demo de Tablero de cotizaciones”	

Prototipo de interface:


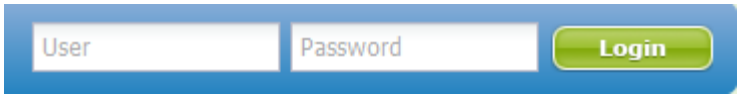
A continuación se muestra en la **Tabla 16**, la descripción de la tarea de ingeniería que se realiza para lograr el cumplimiento de la historia de usuario anterior:

Tabla 16: Tarea de Ingeniería Registrar usuario.

Tarea de Ingeniería	
Número Tarea: 7.1	**Número Historia de Usuario:** HU_7
Nombre Tarea: Registrar usuario.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 30/01/2012	**Fecha Fin:** 10/02/2012
Programador Responsable: Roylandi González Pujol	
Descripción: Desarrollar la opción de registrar usuario que permite acceder a la aplicación e interactuar con la misma.	

Tabla 17: Historia de Usuario Autenticar usuario.

Historia de Usuario	
Número:HU_8	**Nombre Historia de Usuario:** Autenticar usuario.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roylandi González Pujol	**Iteración Asignada:**3
Prioridad en Negocio: Alta	**Puntos Estimados:**2
Riesgo en Desarrollo: Muy *Alto*	**Puntos Reales:** 2

Descripción: La presente historia de usuario tiene como objetivo permitir la autenticación del usuario en el sistema para que pueda acceder a la aplicación e interactuar con la misma.
Observaciones: Para que esto sea posible el usuario debe dirigirse al “Demo de Tablero de cotizaciones” que se encuentra en la pestaña “Demos” del sitio de jWebSocket. Debe autenticarse en el demo de “Demo de Tablero de cotizaciones”
Prototipo de interface: 

A continuación se muestra en la **Tabla 18**, la descripción de la tarea de ingeniería que se realiza para lograr el cumplimiento de la historia de usuario anterior:

Tabla 18: Tarea de Ingeniería Autenticar usuario.

Tarea de Ingeniería	
Número Tarea: 8.1	**Número Historia de Usuario:** HU_8
Nombre Tarea: Autenticar usuario.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 13/02/2012	**Fecha Fin:** 23/02/2012
Programador Responsable: Roylandi González Pujol	
Descripción: Desarrollar la opción de autenticar usuario que permite acceder a la aplicación e interactuar con la misma.	

2.6 Plan de Release

El Plan de Release permite 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 de usuario se llegó a la siguiente planificación la cual se muestra en la **Tabla 19:**

Tabla 19: Plan de Release

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
Iteración 2	En esta iteración se va a desarrollar una primera versión de la aplicación para la cual se implementarán todas las historias de usuario que constituyen funcionalidades fundamentales de la aplicación. Estas tienen prioridad muy alta.	HU_1 HU_2 HU_3 HU_4 HU_5 HU_6	12 semanas
Iteración 3	En esta iteración se va a desarrollar una segunda versión de la aplicación. Se implementarán las historias de usuario para gestionas usuarios teniendo prioridad alta.	HU_7 HU_8	1 semana

2.7 Arquitectura de la solución

La arquitectura de una aplicación es la vista conceptual de su estructura. Toda aplicación contiene código de presentación, de procesamiento y de almacenamiento de datos. La arquitectura de las aplicaciones difiere según como esté distribuido su código.

Para desarrollar la aplicación Stock Ticker Demo se utiliza una arquitectura en capas. La programación por capas es una arquitectura cliente-servidor la cual tiene como objetivo primordial la separación de la lógica de negocios y la lógica de diseño. La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y en caso de que ocurra algún cambio sólo afectará dicho nivel, logrando obviar las demás capas del sistema. Permite además distribuir el

trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles. A continuación se muestra en la **Fig. 3** el diagrama de arquitectura de la aplicación Stock Ticker Demo para la gestión de datos financieros mediante un tablero de cotizaciones en tiempo real.

Fig. 3: Arquitectura de la solución. Fuente: Elaboración propia.

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.

Se representa en la **Fig. 4** el diagrama de paquetes de la aplicación Stock Ticker Demo que se propone:

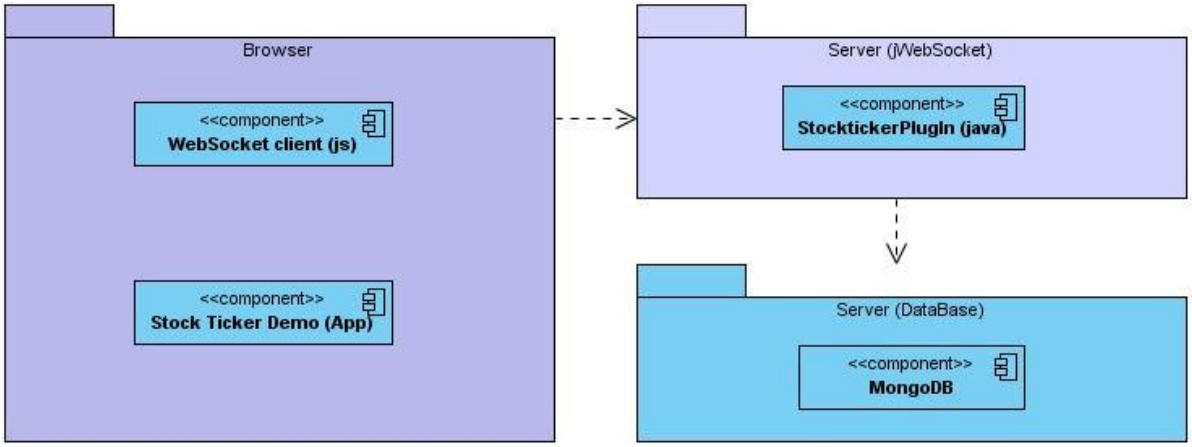


Fig. 4: Diagrama de Paquetes de la aplicación Stock Ticker Demo. Fuente: Elaboración propia.

A continuación se describen los principales elementos del diagrama de paquetes:

- ✓ En la capa del browser se tienen los componentes jWebSocket Client y Stock Ticker Demo. jWebSocket Client constituye el cliente del marco de trabajo jWebSocket y Stock Ticker Demo representa la aplicación. Esta capa del browser se relaciona con la capa del servidor de jWebSocket.
- ✓ La capa del servidor de jWebSokcet contiene el plugin Stockticker PlugIn de la aplicación el cual se encarga de proveer al cliente la información y se relaciona con la capa del servidor de Base de Datos.
- ✓ La capa del servidor de Base de datos contiene la base de datos de MongoDB el cual interactúa con el servidor de jWebSocket, permitiendo persistir o consultar los datos de la aplicación.

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 en la **Fig. 5** el diagrama de componentes para la

solución que se propone:

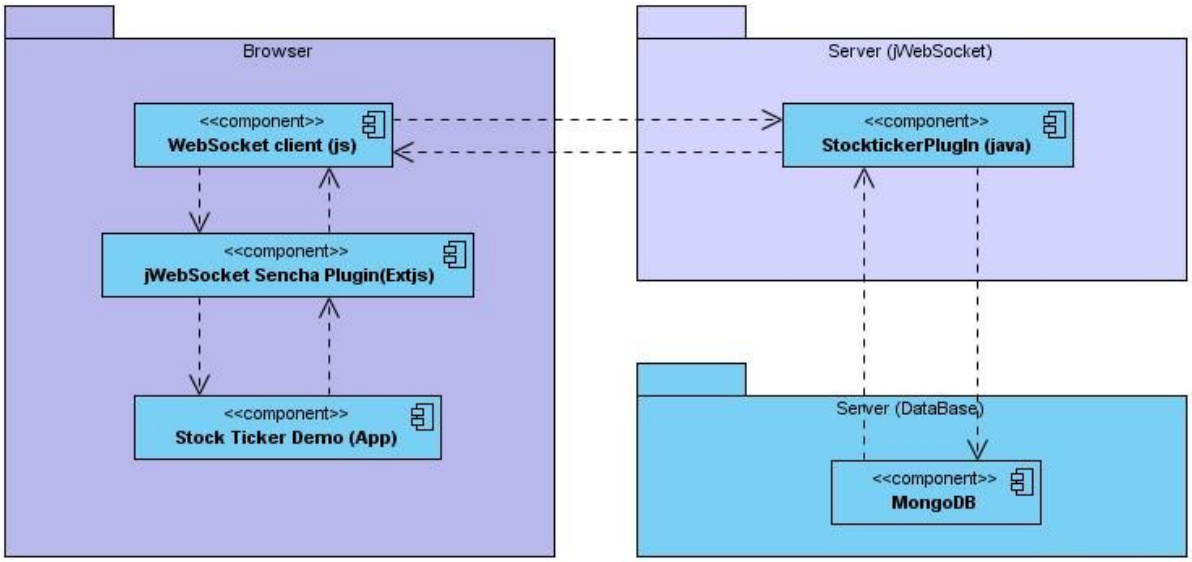


Fig. 5: Diagrama de Componentes de la aplicación Stock Ticker Demo. Fuente: Elaboración propia.

- ✓ Los componentes son los *plug-ins* o filtros de las aplicaciones.
- ✓ El paquete de Browser contiene componentes con los que debe interactuar el cliente.
- ✓ El paquete Server (jWebSocket) contiene el componente que interactúan en el servidor.
- ✓ El paquete Server (Data Base) contiene el componente MongoDB, el cual interactúa con el servidor para proporcionar los datos persistentes.

Conclusiones del Capítulo

En este capítulo se definieron las características y funcionalidades del sistema desarrollado. Quedaron aprobados los requisitos funcionales necesarios para obtener un sistema eficiente. Se presenta la aplicación Stock Ticker Demo como propuesta de solución para la representación gráfica de datos en tiempo real utilizando el marco de trabajo jWebSocket.

CAPÍTULO 3. IMPLEMENTACIÓN Y VALIDACIÓN DEL SISTEMA

Introducción

En el presente capítulo queda documentada la implementación de la solución propuesta en el capítulo anterior, generándose el diagrama de despliegue de la aplicación Stock Ticker Demo. Además se exponen las pruebas funcionales a las 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 muestra la estrategia de validación diseñada en la investigación para comprobar la funcionalidad de la aplicación Stock Ticker Demo y la incidencia de la misma en el aumento de la usabilidad de marco de trabajo jWebSocket en el desarrollo de aplicaciones de gestión financiera en la Web.

3.1 Implementación

La aplicación Stock Ticker Demo está desarrollada sobre el marco de trabajo jWebSocket. La implementación de esta aplicación está dividida en tres partes una para el lado del cliente disponible solamente para el lenguaje JavaScript otra para el servidor en el lenguaje Java, y la última la de acceso a datos utilizando como lenguaje Java y el gestor de base de datos documental MongoDB.

A continuación se muestra en la **Fig. 6** una descripción del cliente de la aplicación Stock Ticker Demo.

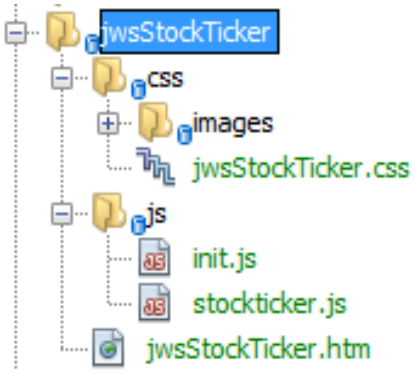


Fig. 6: Descripción del cliente de la aplicación Stock Ticker Demo. Fuente: Elaboración propia.

En el lado del cliente se hace uso de los siguientes archivos JavaScript:

- ✓ **stockticker.js:** Permite representar gestionar los datos que llegan del servidor.
- ✓ **Init.js:** Es en encargado de inicializar la aplicación.

Se hace uso además de archivos htm y css para la interfaz de usuario de la aplicación:

- ✓ **jwsStockTicker.html:** En este archivo está definido todo el código HTML de la aplicación.
- ✓ **jwsStockTicker.css:** Incluye el css correspondiente al HTML de la aplicación.

Los métodos más significativos en el lado del cliente se describen a continuación en la **Tabla 20**.

Tabla 20: Métodos Principales del lado del cliente.

Método	Descripción
getWinLog()	Crea la ventana de inicio que gestiona el registro y autenticación del usuario
getWinTicker()	Crea la ventana principal donde se muestran y gestionan los datos por el usuario.
getWinBuy()	Crea la ventana de comprar donde se gestionan las compras hechas por usuario.
getWinSell()	Crea la ventana de vender donde se gestionan las ventas hechas por usuario.
getWinHistory()	Crea la ventana muestra los últimos 10 valores del precio de la empresa seleccionada por el usuario.
login()	Se encarga de enviar al servidor el usuario autenticado, si es erróneo lanza un mensaje de error, en caso contrario el usuario accede a la ventana principal.
register()	Se encarga de enviar al servidor el usuario registrado, si es erróneo lanza un mensaje de error, en caso contrario lanza un mensaje satisfactorio.

En el lado del servidor la aplicación Stock Ticker Demo hace uso de la estructura de paquetes que se observa en la **Fig. 7:**

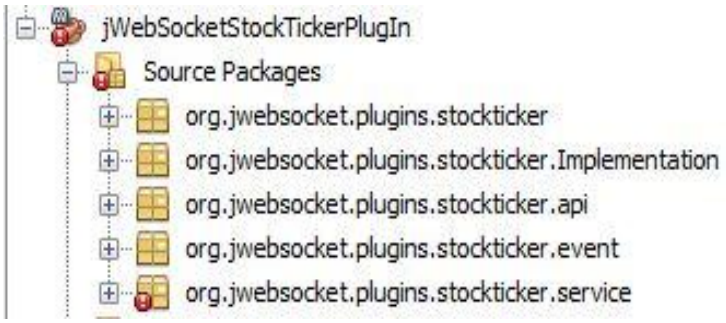


Fig. 7: Descripción del servidor de la aplicación Stock Ticker Demo. Fuente: Elaboración propia.

- ✓ org.jwebsocket.plugin.stockticker: Contiene el plug-in que implementa todas las funcionalidades de la aplicación.
- ✓ org.jwebsocket.plugin.stockticker.Implementation: Contiene las implementaciones de las interfaces.
- ✓ org.jwebsocket.plugin.stockticker.api: Contiene las interfaces que van a ser implementadas por la aplicación.
- ✓ org.jwebsocket.plugin.stockticker.event: Contiene las implementaciones de los eventos de la aplicación.
- ✓ org.jwebsocket.plugin.stockticker.service: Contiene la implementación de los servicios de la aplicación.

Los métodos más significativos en el lado del servidor se describen a continuación en la **Tabla 21**.

Tabla 21: Métodos Principales del lado del cliente.

Método	Descripción
createUser(IUser aUser)	Almacena en la base de datos los usuarios registrados.
login(IUser aUser)	Verifica que el usuario exista en la base de datos.
valuesOfRecords(Integer aId, String aName)	Genera los valores aleatorios para ser mostrados en el tablero de cotizaciones.
listRecords()	Obtiene una lista de empresas con los valores de la base de datos.
buy(String aName, String aCant, String aUserLogin)	Almacena en la base de datos las compras de instrumentos hechas por el usuario.

sell(String aName, String aCant, String aUserLogin)	Almacena en la base de datos las ventas hechas por el usuario.
readBuy(String aUser)	Obtiene una lista por usuario del estado de su depósito en la base de datos.
showComb(String aUser)	Obtiene una lista por usuario del nombre de las empresas de su depósito en la base de datos.
chart(String aUserLogin, String aName)	Obtiene por usuario el valor de la cantidad del nombre de las empresas de su depósito en la base de datos.

Diseño de la Base de Datos

Para la persistencia de los datos financieros que se gestionan en la aplicación Stock Ticker Demo se diseña una base de datos documental. A continuación se muestra en la **Fig. 8** el diagrama entidad-relación de la base de datos diseñada.

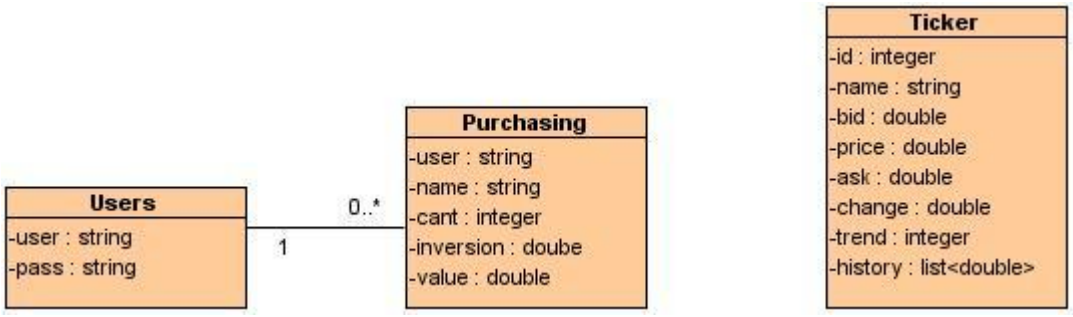


Fig. 8: Diagrama entidad-relación. Fuente: Elaboración propia.

En la **Tabla 22, Tabla 23** y **Tabla 24** respectivamente son descritas de manera detallada cada una de colecciones que conforman la base de datos de la aplicación Stock Ticker Demo.

Tabla 22: Colección Users

Entidad:	users		
Descripción:	Esta colección va a contener los datos de los usuarios del sistema.		
Relaciones:	purchasing		
Campos	**Tipo de Dato**	**Tamaño**	**Descripción**
user	String	-	El user va a ser un nombre escogido por el usuario, y constituye la llave primaria de esta colección.
pass	String	-	Contraseña del usuario

Tabla 23: Colección Ticker

Entidad:	ticker		
Descripción:	Esta entidad va a contener las acciones realizadas por cada uno de los usuarios del sistema.		
Relaciones:			
Campos	**Tipo de Dato**	**Tamaño**	**Descripción**
name	string	-	Nombre del instrumento de la colección.
bid	double	-	Valor de la oferta para comprar un instrumento.
price	double	-	Valor del precio del instrumento.
ask	double	-	Valor de la oferta para vender el instrumento.
change	double	-	Valor en porciento de la diferencia del price actual con el anterior.
trend	boolean	-	Valor en dependencia del precio.
history	List<double>	-	Listado de los últimos diez price que tomo el instrumento.

Tabla 24: Colección Purchasing

Entidad:	purchasing		
Descripción:	Esta entidad va a contener las acciones realizadas por cada uno de los usuarios del sistema.		
Relaciones:	user		
Campos	**Tipo de Dato**	**Tamaño**	**Descripción**
user	string	-	Usuario que está haciendo las compras.
name	string	-	Nombre del instrumento comprado en la colección ticker.
cant	integer	-	Cantidad de acciones de un instrumento comprado.
inversión	double	-	Cantidad de dinero usado en las compras
value	double	-	Valor que van tomando los instrumentos.

3.2 Diagrama de Despliegue

El diagrama de despliegue es un artefacto que modela la arquitectura en tiempo de ejecución de un sistema. El diagrama de despliegue es el complemento del diagrama de componente que, unidos, proveen la vista de implementación del sistema. Se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. 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 en la **Fig. 9** el diagrama de despliegue, que representa la distribución física de la aplicación Stock Ticker Demo 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.

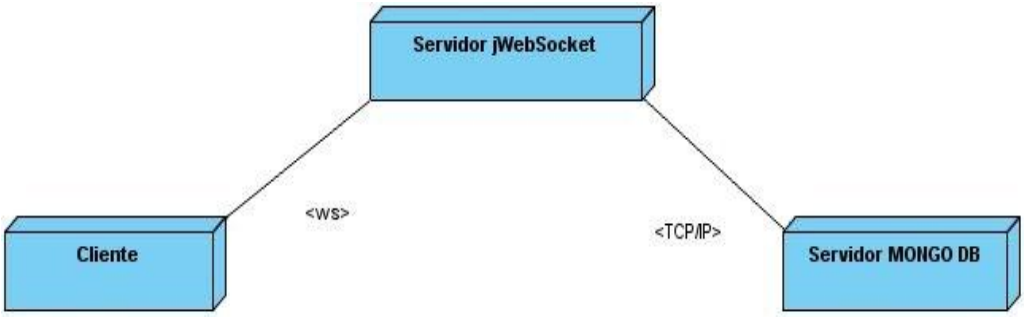


Fig. 9: Diagrama de Despliegue de la solución. Fuente: Elaboración propia.

Se describen a continuación cada uno de los componentes del Diagrama de Despliegue de la aplicación.

- ✓ **Cliente:** El cliente se conectará mediante el protocolo WebSocket al Servidor jWebSocket para lo cual solo necesitará un navegador que soporte WebSocket sobre cualquier sistema operativo.
- ✓ **Servidor jWebSocket:** El servidor de jWebSocket se conectará al servidor de base de datos de MongoDB a través del protocolo TCP/IP para persistir o consultar los datos.

- ✓ **Servidor MongoDB:** Servidor de base de datos el cual interactúa con el servidor de jWebSocket para proporcionar los datos persistentes.

3.3 Validación de la investigación

Para la validación de la presente investigación se traza una estrategia con el objetivo de medir el impacto de la aplicación Stock Ticker Demo en la usabilidad del marco de trabajo jWebSocket en el desarrollo de tableros de cotizaciones en tiempo real. La estrategia está conformada por tres fases, la primera consiste en realizar pruebas funcionales a la aplicación por parte de equipo de desarrollo. Seguidamente se realiza un proceso de certificación de calidad de software por el grupo de calidad del Centro de Desarrollo de la Facultad Regional “Mártires de Artemisa”. Este grupo como agente externo al proyecto realiza un proceso de certificación basados en diferentes elementos funcionales y documentales, de acuerdo a la metodología SXP seleccionada para el desarrollo de la aplicación Stock Ticker Demo.

Durante la tercera etapa de la estrategia de validación se pone a valoración del cliente la aplicación desarrollada. El cliente de la aplicación Stock Ticker Demo es Alexander Schulze fundador y líder de proyecto jWebSocket. Esta etapa de validación tiene como objetivo lograr que el cliente emita su valoración sobre la aplicación desarrollada, su integración al marco de trabajo de manera internacional y la usabilidad de la aplicación para los usuarios potenciales y desarrolladores de la comunidad de jWebSocket. En los epígrafes siguientes se detallan cada una de las etapas de la estrategia de validación de la presente investigación.

3.4 Casos de Pruebas

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

todas las historias de usuario se definieron casos de prueba. A continuación se muestran en las **Tabla 25**, **Tabla 26**, **Tabla 27** y **Tabla 28** los casos de pruebas realizados. En el Anexo 1 se observan los restantes casos de pruebas.

Tabla 25: Caso de prueba para la HU Mostrar tablero de cotizaciones.

Caso de Prueba Funcional	
Código Caso de Prueba:jws-1-1	**Nombre Historia de Usuario:** Mostrar tablero de cotizaciones
Nombre de la persona que realiza la prueba: Roylandi González Pujol	
Descripción de la Prueba: Esta prueba consiste en verificar que se pueda mostrar el tablero de cotizaciones con cada una de las empresas y sus datos correspondientes.	
Condiciones de Ejecución: Para poder realizar la prueba se tiene que utilizar un navegador que soporte el protocolo WebSocket y el servidor de la aplicación tiene que estar ejecutándose.	
Entrada / Pasos de ejecución: Para mostrar el tablero de cotizaciones se debe de haber autenticado el usuario en la aplicación.	
Resultado Esperado: Se muestran los datos mediante una tabla.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 26: Caso de prueba para la HU Mostrar historial.

Caso de Prueba Funcional	
Código Caso de Prueba:jws-2-1	**Nombre Historia de Usuario:** Mostrar historial.
Nombre de la persona que realiza la prueba: Roylandi González Pujol	
Descripción de la Prueba: Esta prueba consiste en verificar que se pueda mostrar el historial de los últimos 10 precios de cada una de las empresas.	

Condiciones de Ejecución: Para poder realizar la prueba se tiene que utilizar un navegador que soporte el protocolo WebSocket y el servidor de la aplicación tiene que estar ejecutándose.
Entrada / Pasos de ejecución: Para mostrar el historial los últimos 10 precios que va tomando una empresa se debe dar doble click encima de la fila del instrumento que quiere ver su historial.
Resultado Esperado: Se muestran los datos mediante una tabla.
Evaluación de la Prueba: Satisfactoria.

Tabla 27: Caso de prueba para la HU Comprar instrumentos.

Caso de Prueba Funcional	
Código Caso de Prueba:jws-3-1	**Nombre Historia de Usuario:** Comprar instrumentos.
Nombre de la persona que realiza la prueba: Roylandi González Pujol	
Descripción de la Prueba: Esta prueba consiste en verificar que se pueda comprar un instrumento del tablero de cotizaciones	
Condiciones de Ejecución: Para poder realizar la prueba se tiene que utilizar un navegador que soporte el protocolo WebSocket y el servidor de la aplicación tiene que estar ejecutándose.	
Entrada / Pasos de ejecución: Durante el proceso de compra de instrumento de la tabla de cotizaciones el usuario debe de estar autenticado en la aplicación, dar click en el botón “Buy Stock” que se encuentra en la parte inferior izquierda del tablero de cotizaciones en el centro de la aplicación para acceder al formulario de vender instrumento. Una vez accedido al formulario se deberán llenar los campos “InstrumentName” y “Amount Stocks” y luego debe hacer clic en el botón “Buy”.	
Resultado Esperado: Se muestran los datos mediante una tabla.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 28: Caso de prueba para la HU Comprar instrumentos.

Caso de Prueba Funcional	
Código Caso de Prueba:jws-3-2	**Nombre Historia de Usuario:** Comprar instrumentos.
Nombre de la persona que realiza la prueba: Roylandi González Pujol	
Descripción de la Prueba: Esta prueba consiste en verificar que cuando el usuario comete un error durante el proceso de compra de un instrumento, se le emite un mensaje de error.	
Condiciones de Ejecución: Para poder realizar la prueba se tiene que utilizar un navegador que soporte el protocolo WebSocket y el servidor de la aplicación tiene que estar ejecutándose.	
Entrada / Pasos de ejecución: Durante el proceso de compra de instrumento de la tabla de cotizaciones el usuario debe de estar autenticado en la aplicación, dar click en el botón “Buy Stock” que se encuentra en la parte inferior izquierda del tablero de cotizaciones en el centro de la aplicación para acceder al formulario de vender instrumento. Una vez accedido al formulario el usuario deja vacío el campo “InstrumentName” o introduce en el campo “Amount Stocks” una cantidad que sumada a la de su depósito exceda a 100, luego debe hacer clic en el botón “Buy”.	
Resultado Esperado: La aplicación muestra un mensaje de error informando que hay un error en la compra del instrumento.	
Evaluación de la Prueba: Satisfactoria.	

3.5 Certificación de Calidad del Software

Para dar la certificación de calidad del software se puso a consideración del grupo de calidad del Centro de Desarrollo de la Facultad el proyecto de software con los artefactos generados por la metodología SXP. Los artefactos son: Lista de Reserva del Producto (LRP), Modelo de Historia de Usuario del Negocio, Historia de Usuario, Arquitectura de Software, Modelo Canónico de Datos, Tarea de Ingeniería,

Plan de Release, Estándar de Código, Caso de Prueba Funcional, Manual de Usuario, Manual de Instalación, Manual de desarrollo. También se puso a consideración del grupo de calidad el código fuente de la aplicación Stock Ticker Demo. Este grupo realizó la revisión del código fuente teniendo en cuenta la estandarización, estructuración y limpieza del mismo. El proceso de revisión de los artefactos se llevó a cabo por la Ingeniera Maidel Ojeda Cruz, asesora de calidad del Centro de Desarrollo y el proceso de revisión del código fuente fue desarrollado por el Ingeniero Domma Moreno Dager, asesor de tecnología. Se garantiza de esta manera la funcionalidad de la aplicación Stock Ticker Demo y la calidad de documentación generada durante el proceso de desarrollo.

3.6 Valoración del Cliente

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

3.7 Resultados Obtenidos

La aplicación Stock Ticker Demo fue sometida a un conjunto de pruebas funcionales las cuales demuestran el cumplimiento de las funcionalidades que debe cumplir la aplicación.

A través de la certificación de calidad realizada por terceros se comprobó que tanto la documentación como el código cuentan con la calidad, limpieza y estandarización que se necesita. Esto brinda la posibilidad que otros usuarios lo puedan usar y asegura además su disponibilidad ante futuras mejoras. El aval emitido por el grupo de calidad se encuentra en el Anexo 2.

El criterio emitido por el cliente permite que la aplicación va a ser utilizada por la comunidad de jWebSocket e integrado con el proyecto internacional jWebSocket. El

certificado emitido por Alexander Schulze se encuentra en el Anexo 3.

Al obtener resultados satisfactorios queda disponible la versión 1.0 de la aplicación Stock Ticker Demo. Se logró una aplicación que cumple con todas las especificaciones del cliente para permitir la gestión y visualización de datos en tiempo real, permitiendo observar y analizar eficientemente la información.

3.8 Funcionalidades Obtenidas

Entre las principales funcionalidades que posee la aplicación Stock Ticker Demo en su versión 1.0 se pueden mencionar:

- ✓ Permite mostrar mediante un tablero de cotizaciones el estado de las acciones de las empresas.
- ✓ Permite mostrar el historial de los últimos 10 precios que va tomando el valor de las acciones de una empresa.
- ✓ Permite comprar instrumentos del tablero de cotizaciones.
- ✓ Permite vender instrumentos de la tabla de depósitos.
- ✓ Permite representar gráficamente los depósitos de compra por usuario en intervalos de tiempo.
- ✓ Permite mostrar el depósito de las compras y ventas de los usuarios.

3.9 Aporte Social y Económico

La aplicación Stock Ticker Demo muestra mediante un tablero de cotizaciones en tiempo real la capacidad del marco de trabajo jWebSocket para el desarrollo de aplicaciones de gestión financiera. La aplicación permite a un usuario interesado en la compra de acciones de una determinada empresa puede consultar datos de interés. Con esta información el usuario decide qué acciones desea comprar de acuerdo a los precios de compra y venta alcanzados de cada acción.

Al observar las cotizaciones el usuario puede ver el momento exacto en que los precios de las acciones empiezan a caer y puede utilizar ese momento para comprar sus acciones. Esto le permite comprar en el momento más ventajoso. Por el contrario, si desea vender acciones, simplemente debe esperar hasta que los

precios comienzan a desplazarse hacia arriba, antes de vender.

Las funcionalidades de la aplicación Stock Ticker Demo de esta manera permite a los desarrolladores internacionales identificar al marco jWebSocket como una herramienta potente para brindar a los usuarios una eficiente gestión financiera. De esta manera la comunidad jWebSocket amplía su alcance y gana colaboradores a nivel internacional vinculados a aspectos económicos y financieros.

Este hecho tiene una influencia económica que beneficia al proyecto jWebSocket además. Aumentar los niveles de uso del marco de trabajo garantiza aumento de las utilidades del proyecto en los productos comerciales de la comunidad. Al lograr mayor satisfacción de los usuarios logra que jWebSocket alcance mayores cuotas de mercados a nivel internacional.

Conclusiones del Capítulo

Al concluir el presente capítulo se definió el diagrama de despliegue reflejando la situación física de los componentes desarrollados. Se presentó la estrategia de validación a través de las pruebas funcionales, la certificación de calidad de software y valoración dada por el cliente de la aplicación. Queda demostrada así que la aplicación Stock Ticker Demo se encuentra en óptimas condiciones para mostrar y gestionar datos en tiempo real en tableros de cotizaciones en la Web.

CONCLUSIONES

Al realizar la presente investigación se realizó un profundo estudio acerca de los fundamentos teórico-metodológicos, el cual aportó elementos significativos que posibilitaron definir que un tablero de cotizaciones permite que las empresas reflejen sus finanzas, introduzcan órdenes y realicen negociaciones de compra y venta de valores mediante una aplicación. Al analizar las tendencias actuales de las tecnologías utilizadas en la Web para desarrollar tableros de cotizaciones se identificaron herramientas que utilizan el protocolo HTTP y otras que ya implementan la comunicación con WebSocket.

jWebsocket es uno de los marcos de trabajos que garantiza tableros de cotizaciones en tiempo real, sin embargo en la presente investigación se identifica que no posee una vía directa que muestre sus potencialidades a los desarrolladores. Esto condujo al desarrollo de la aplicación demostrativa Stock Ticker Demo que gestiona y visualiza los datos de un tablero de cotizaciones en tiempo real utilizando el marco de trabajo jWebSocket.

La aplicación fue sometida a una estrategia de validación compuesta por pruebas funcionales, certificación de calidad de software y valoración del cliente con el fin de comprobar su funcionamiento. El resultado de la investigación muestra que la aplicación Stock Ticker Demo constituye hoy día una vía para mostrar las potencialidades de jWebSocket en la gestión financiera de datos en tiempo real y posibilita de esta manera aumentar los niveles de usabilidad del este marco en este tipo de aplicaciones.

RECOMENDACIONES

Desarrollar una segunda versión de la aplicación Stock Ticker Demo incluyéndole nuevas funcionalidades que aumenten las potencialidades del proceso de gestión de datos de tableros de cotizaciones. Además se recomienda integrar esta aplicación a algunas de las bases de datos internacionales. Las más reconocidas son NYSE, NASDAQ y AMEX para que sea usada por los corredores de bolsas y empresas internacionales.

BIBLIOGRAFÍA REFERENCIADA

¿Que es Mootools? **Garcia, Pablo. 2009.** s.l. : http://comoganardinerodesdecasa.es/%C2%BFque-es-mootools/, 2009.

¿Qué es un 'framework'? **Sánchez, Jordi. 2006.** Valencia : http://jordisan.net/blog/2006/que-es-un-framework, 2006.

Apache Software Foundation. 2011. Apache Subversion. *Apache Subversion.* [En línea] 10 de 12 de 2011. [Citado el: 10 de 12 de 2011.] http://subversion.apache.org/.

Carga de clases dinámicamente con ExtJS 4. **Villa, Crysfel. 2011.** s.l. : http://www.quizzpot.com/2011/10/carga-de-clases-dinamicamente-con-extjs-4/, 2011.

Comparación de Frameworks en Javascript. **Tavárez, David. 2009.** s.l. : http://www.maestrosdelweb.com/editorial/comparacion-frameworks-javascript/, 2009.

Dirección de Supervisión, Fiscalización y Estudios. Sub Dirección de Estudios Económicos y de Mercado. 2010. *Bienes sujetos a cotización internacional o cuyo precio está influido por ésta. ¿Cuáles son o cómo delimitarlos? Análisis del artículo 49º del Reglamento de la Ley de Contrataciones del Estado.* 2010.

Eclipse. 2011. *The Eclipse Foundation Open Source Community Website.* [En línea] Eclipse Foundation, 2011. http://www.eclipse.org/.

Eguíluz Pérez, Javier. 2009. *CSS avanzado.* http://www.librosweb.es/css_avanzado/capitulo5/el_framework_yui.html : s.n., 2009.

Erika Dolores Baez Montero, Alfredo Iván Islas Domínguez. 2009. *Reingeniría en el proceso de ventas para el manejo de cotizaciones en SlySTEC S.A de C.V.* Mexico : s.n., 2009.

Extjs. 2009. www.extjs.es. [En línea] 2009. [Citado el: 10 de 12 de 2011.]

http://www.extjs.es.

Fernández, Mónica Merchán. 2008. *Análisis del cambio organizacional y la función financiera.* 2008.

Figueroa, Roberth G, Solís, Camilo J y Cabrera, Armando A. 2011. Entorno Virtual de Aprendizaje. [En línea] 2011. [Citado el: 13 de 12 de 2011.] http://eva.uci.cu/mod/resource/view.php?id=9303&subdir=/Metodologias/RUP.

Framework Approach for WebSockets. **Schulze, Alexander. 2011.** Web Technologies & Internet Applications (WebTech 2011) : http://dl.globalstf.org/index.php?page=shop.product_details&flypage=flypage_images.tpl&product_id=528&category_id=42&option=com_virtuemart&Itemid=4&vmcchk=1&Itemid=4, 2011.

Gijón, Rosa Maria Domínguez. 2010. *Aplicación de la teoría de valores extremos para analizar el comportamiento del índice de precios y cotizaciones en el mercado de valores Mexicano.* Mexico : s.n., 2010.

Grupo Soluciones Innova. [En línea] [Citado el: 11 de 12 de 2011.] http://www.rational.com.ar/herramientas/roseenterprise.html.

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

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

jQuery, la librería Javascript por excelencia. Un framework Javascript lleno de ventajas. **Soriano, Javier. 2011.** Valencia : http://www.dicreato.com/blog/jquery-la-libreria-javascript-por-excelencia-un-framework-javascript-lleno-de-ventajas/, 2011.

Murphey, Rebecca. 2010. *jQuery Fundamentals.* s.l. : Autoedición, 2010.

NetBeans. 2011. *NetBeans.* [En línea] Oracle Corporation, 2011. http://netbeans.org/features/index.html.

Nourie, Dana. 2005. *Oracle Sun Developer Network (SDN).* [En línea] Oracle

Corporation, 24 de Mayo de 2005. http://java.sun.com/developer/technicalArticles/tools/intro.html.

Proietti, Valerio. 2009. *MooTools.* [En línea] 2009. [Citado el: 14 de 12 de 2011.] http://mootools.net/.

Prototype Core Team. 2007. Prototype. [En línea] 2007. [Citado el: 14 de 12 de 2011.] http://www.prototypejs.org/.

Proyecto jQuery. 2010. License. *jQuery.* [En línea] 2010. [Citado el: 8 de 12 de 2011.] http://jquery.org/license/.

Qué es jQuery, para qué sirve y qué ventajas tiene el utilizar este framework Javascript. **Alvarez, Miguel Angel. 2009.** s.l. : http://www.desarrolloweb.com/articulos/introduccion-jquery.html, 2009.

RapidSVN. 2011. RapidSVN. [En línea] 2011. [Citado el: 13 de 12 de 2011.] http://www.rapidsvn.org/.

Schwaber, Ken y Beedle, Mike. 2011. *Agile Software Development with SCRUM.* s.l. : Prentice Hall, 2011. 0130676349.

Sencha. 2011. Ext JS 4 JavaScript Framework for Rich Apps in Every Browser. *Sencha.* [En línea] 2011. [Citado el: 14 de 12 de 2011.] http://www.sencha.com/products/extjs/.

Surhone, Lambert M, T Tennoe, Mariam y Henssonow, Susan F. 2010. *Real-Time Web.* s.l. : VDM Verlag, 2010. 9786133432239.

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

SXP, METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE. **Peñalver, G, Meneses, A y García, S. 2010.** Chile : 1er congreso Iberoamericano de Ingeniería de Proyectos, 2010.

team, TortoiseSVN. 2011. TortoiseSVN. [En línea] 2011. [Citado el: 13 de 12 de

2011.] http://tortoisesvn.net/.

Team, TortoiseSVN. 2011. TortoiseSVN. [En línea] 2011. [Citado el: 13 de 12 de 2011.] http://tortoisesvn.net/.

Visual Paradigm. [En línea] [Citado el: 11 de 12 de 2011.] http://www.visual-paradigm.com.

Wells, Don. 2009. Extreme Programming: A gentle introduction. *Extreme Programming.* [En línea] 28 de 07 de 2009. [Citado el: 16 de 02 de 2012.] http://www.extremeprogramming.org/.

Wordpress.com. 2007. Librería ExtJs. *Desarrollo WEB y otras Hierbas.* [En línea] 06 de 08 de 2007. [Citado el: 13 de 12 de 2011.] http://vargasti.wordpress.com/2007/08/06/libreria-extjs/.

Yahoo! Inc. 2011. YUI Library. *Yahoo!* [En línea] 2011. [Citado el: 14 de 12 de 2011.] http://developer.yahoo.com/yui/.

YUI es un framework para desarrollo de webs que contiene librerías Javascript y CSS para crear interfaces de usuario típicas de los sitios de contenido enriquecido. **Alvarez, Miguel Angel. 2010.** s.l. : http://www.desarrolloweb.com/articulos/libreria-yui-yahoo.html, 2010.