

**Universidad de las Ciencias Informáticas
Facultad Regional Mártires de Artemisa**



**Entorno de Integración Continua para el marco de trabajo
jWebSocket**

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

Autor: Carlos Alberto Feyt Salgueiro

Tutores:

Msc. Margarita González Ferrer

Msc. Yamila Vigil Regalado

Lic. Gilberto Ramón Justiniani Fernández

Ciudad de la Habana, Cuba

Junio 2012

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Desarrollo de la Facultad Regional “Mártires de Artemisa” de la Universidad de las Ciencias Informáticas; así como a 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 _____.

Carlos Alberto Feyt Salgueiro

Firma del Autor

Msc. Margarita González Ferrer

Firma del Tutor

Msc. Yamila Vigil Regalado

Firma del Autor

Lic. Gilberto Ramón Justiniani Fernández

Firma del Tutor

AGRADECIMIENTOS

Le agradezco a toda mi familia por el apoyo incondicional que me han brindado en estos años. Principalmente a mis padres y mi hermana que siempre han estado a mi lado apoyándome y dándome todo su cariño.

También agradezco a mis amistades que me han acompañado en estos 5 años de carrera con los cuales he pasado los mejores momentos de mi vida.

Por último agradezco también a los profesores que me han ayudado durante mi estancia en la universidad.

DEDICATORIA

A mis padres y a mi hermana que siempre han estado ayudándome para poder hacer realidad este sueño y fueron mi principal motivo de inspiración. Gracias a ellos he podido realizar mis sueños como estudiante. A mis padres les prometo que les dedicaré mi futuro y trataré de complacerlos en todo lo que pueda, ya que ese ha sido siempre mi principal objetivo por el que siempre me he superado.

RESUMEN

A lo largo de la historia de la informática, el hombre ha buscado la manera de mejorar los procesos de desarrollo de software. Con la aparición del internet en los años noventa hubo un gran impacto en la industria del software, dando paso al surgimiento de muchas comunidades de desarrollo de software colaborativo. Un ejemplo de comunidad de software colaborativo es proyecto jWebSocket, el cual tiene mucho desarrolladores de diversas partes del mundo.

En la actualidad existen problemas de integración de los códigos de los desarrolladores de proyectos realizados con el marco de trabajo jWebSocket, debido a que este proceso es realizado de forma manual, esto trae como consecuencia poca generación de versiones, poca realización de pruebas al código y mucho tiempo perdido por parte de los desarrolladores corrigiendo errores en el código. Todo esto atenta a la entrega en tiempo y la calidad de los proyectos.

El presente estudio fue realizado con los proyectos desarrollados actualmente en la Facultad Regional Mártires de Artemisa en el período 2011-2012, con el objetivo de proponer un entorno de Integración Continua para el desarrollo de proyectos con el marco de trabajo jWebSocket que permita una mejor calidad del código, menos tiempo de finalización del producto y una frecuente generación de versiones.

Como propuesta de solución se crea un entorno de integración continua compuesto por las siguientes herramientas: JUnit, Subversion, Maven, Apache Archiva, Sonar y Jenkins.

Finalmente se valida la propuesta de solución haciendo uso el método de validación por criterio de especialistas obteniendo un resultado positivo.

ÍNDICE

INTRODUCCIÓN.....	1
CAPITULO 1. FUNDAMENTACIÓN TEÓRICA.....	7
1.1. Conceptos fundamentales.....	7
1.2. Estado del Arte	11
1.3. Características del entorno de desarrollo del proyecto jWebSocket.	12
1.4. Tecnologías usadas para el desarrollo de la solución	13
Conclusiones del capítulo	23
CAPITULO 2. PROPUESTA DE UN ENTORNO DE INTEGRACIÓN CONTINUA.	24
2.1. Estado actual del proceso de desarrollo de proyectos con el entorno de jWebSocket.....	24
2.2. Resultados de la encuesta realizada a especialistas de diferentes proyectos de la facultad para la determinación del diagnóstico del problema	25
2.3. Propuesta de solución	27
2.4. Funcionamiento del entorno de integración continua propuesto	27
2.5. Proceso de descarga de la aplicación	28
2.6. Características del entorno de instalación.....	28
2.7. Proceso de instalación	29
2.8. Opciones de configuración	32
2.9. Administración de la aplicación	37
Conclusiones del capítulo	46
CAPITULO 3. VALIDACIÓN DE LA PROPUESTA.....	47
3.1. Método de validación de la propuesta	47
3.2. Selección de los especialistas para la validación	48
3.3. Elaboración de la encuesta para la validación de la solución.....	49

3.4. Resultados de la validación.....	50
3.5. Aporte Social y Económico.....	59
Conclusiones del capítulo	60
CONCLUSIONES.....	61
RECOMENDACIONES	62
BIBLIOGRAFÍA.....	63
ANEXOS	66
GLOSARIO DE TÉRMINOS.....	71

INTRODUCCIÓN

A lo largo de la historia de la informática, el hombre ha buscado la manera de mejorar los procesos de desarrollo de software. La humanidad ha sido testigo de importantes y trascendentales cambios en los aspectos que han colaborado con el desarrollado del software.

La industria de software nació en 1968, cuando IBM decidió separa la producción de equipos del desarrollo de programas para operarlos. En el futuro, la industria iría desarrollándose a través de una serie de olas de crecimientos causado por cambios de plataforma, en los setenta con la aparición de las minicomputadoras, en los ochenta la computadora personal, en los noventa el Internet.

La gestión de proyectos es un punto importante en la industria del software. Proyecto son todas las acciones que deben realizarse para cumplir con una necesidad dada en un plazo temporalizado que tiene inicio y fin. Mediante la gestión de proyecto se puede organizar y administrar recursos (materiales y humanos) de forma que se pueda terminar todo el trabajo requerido en el proyecto dentro del alcance, el tiempo y el costo definido.

El desarrollo colaborativo ha sido un aspecto relevante en el ciclo de vida del desarrollo del software. El software libre es un ejemplo de desarrollo colaborativo, grandes comunidades de desarrollo dan paso a la existencia de importantes sistemas informáticos como son motores de base de datos, la creación de framework¹, sistemas operativos, entre otros.

Una de las principales herramientas que ha dado soporte al desarrollo colaborativo son los sistemas de control de versiones. Mediante estos se puede almacenar los ficheros fuentes de un proyecto, garantizando un sistema de versiones del código fuente. También permite que los integrantes de un proyecto puedan acceder a todas las versiones por la que va pasando el código de un proyecto.

¹ **Framework:** estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación.

El uso de una herramienta de línea de comandos es también un aspecto importante en el desarrollo colaborativo. Estas permiten la gestión y construcción de proyectos, facilita la creación de la estructura de directorio de un proyecto desde cero, como es la creación de sesiones para el código fuente, las pruebas unitarias, ficheros de configuración, etc. Este tipo de herramientas también facilita los comandos para compilar un proyecto. Es capaz de gestionar las dependencias de librerías que se necesiten usar.

Un aspecto importante en el desarrollo de software a distancia es la integración continua. La integración continua es una práctica de desarrollo de software que permite a los miembros de un equipo integrar su trabajo con frecuencia, generalmente cada persona integra al menos una vez al día pudiendo haber múltiples integraciones por día. Cada integración es realizada automáticamente (incluyendo las pruebas) para detectar errores de integración tan pronto como sea posible. Este enfoque conduce a una reducción significativa en los problemas de integración y permite a un equipo desarrollar software cohesivo más rápidamente. (Fowler, 2006)

La combinación de todas estas tecnologías es lo que le permite a la humanidad hoy en día el desarrollo de grandes aplicaciones a distancia. Un ejemplo de colaboración en línea lo tenemos hoy en la Universidad de Ciencias Informáticas (UCI). Es el caso del proyecto jWebSocket². El proyecto está formado por integrantes de diferentes países del mundo como son Alemania, Estados Unidos, La India y en especial Cuba, el cual cuenta con la mayor cantidad de miembros de esta comunidad con más de 50% de participación. (Schulze, 2011)

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

² **jWebSocket:** <https://jwebsocket.org>

abstracción en la manipulación de los diferentes formatos, el framework convierte los paquetes de datos entrantes y salientes en tokens. El cliente nativo soporta el intercambio de paquetes en los formatos JSON³, XML⁴ y CSV⁵ 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. (Framework Approach for WebSockets, 2011)

Debido a los hechos anteriormente planteados surge la siguiente **situación problemática**:

Actualmente existen dificultades en el proceso colaborativo entre los desarrolladores de jWebSocket. Al tener varios miembros del equipo trabajando en un mismo proyecto, se dificulta la integración de los códigos de cada uno de estos integrantes. En estos momentos este proceso es efectuado manualmente por una persona, siendo una tarea tediosa de realizar y que requiere largas jornadas de trabajo. Esto trae como consecuencia una poca agregación de archivos al repositorio y consigo poca generación de versiones. También la realización de pruebas del código solo es efectuada en la finalización de una versión, lo que da paso a la existencia de muchos defectos en el código, atentando contra el tiempo y la calidad de la finalización del proyecto.

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

¿Cómo garantizar la integración continua de código fuente en proyectos de desarrollados con el marco de trabajo de jWebSocket que permita una mejor calidad del código, menos tiempo de finalización del producto y una frecuente generación de versiones?

Por tanto el presente trabajo centra su **objeto de estudio** en la integración continua para el desarrollo de aplicaciones informáticas.

³ **JSON**: (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.

⁴ **XML**: (eXtensible Markup Language - lenguaje de marcas extensible) es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

⁵ **CSV**: (comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla.

Como **Campo de Acción** Sistema de Integración continua para aplicaciones desarrolladas en Java con el marco de trabajo de jWebSocket.

Para dar solución al problema anterior se plantea como **objetivo general**: Proponer un entorno de Integración Continua para el desarrollo de proyectos con el marco de trabajo jWebSocket que permita una mejor calidad del código, menos tiempo de finalización del producto y una frecuente generación de versiones.

Para dar cumplimiento al Objetivo propuesto se formulan las siguientes **preguntas científicas**:

- ¿Cuáles son los fundamentos teóricos de los entornos de Integración Continua?
- ¿Cuál es la situación actual de la integración de los códigos creados por los distintos desarrolladores en el proyecto jWebSocket?
- ¿Cómo proponer un entorno de Integración Continua para aplicaciones Java?
- ¿Cómo validar el entorno de Integración Continua propuesto?

Variable Independiente: Entorno de Integración Continua.

Variables Dependientes: Calidad de código, tiempo de finalización del proyecto y generación de versiones.

Las **Tareas de Investigación** desarrolladas para cumplir los objetivos propuestos son:

- Fundamentación del marco teórico de los entornos de Integración Continua.
- Caracterización de la situación actual del entorno de trabajo en el grupo de desarrollo de jWebSocket.
- Propuesta de un entorno de Integración Continua en aplicaciones Java para el desarrollo de proyectos con el marco de trabajo jWebSocket.
- Validación de la propuesta mediante el criterio de especialistas.

Esta investigación se sustenta en el método dialéctico materialista ya que aplica el conocimiento de leyes, principios y categorías universales para operar en la realidad, partiendo de la posibilidad de aplicar los métodos teóricos y empíricos para asumir con científicidad el objeto de estudio, y contribuir a la transformación práctica de la realidad, en beneficio de los entornos de Integración Continua.

Los **Métodos Científicos** utilizados en esta investigación fueron:

Métodos teóricos

- **Histórico-Lógico:** Permite analizar la trayectoria completa acerca de los entornos de integración continua para aplicaciones Java, así como el estudio histórico de los mismos que permite ver deficiencias y proponer soluciones acorde a las necesidades.
- **Análisis-Síntesis:** Mediante este método se va a analizar toda la teoría recopilada a través de los diferentes medios bibliográficos.
- **Modelación:** Este método permite realizar una representación de la situación que se analiza. Permite obtener mediante diagramas y objetos una mayor comprensión del problema y ayudar a configurar un entorno de integración continua para las aplicaciones Java partir de la situación problemática.

Métodos empíricos

- **Encuesta:** Este método permitió diagnosticar el problema y validar la solución al ser aplicada a los especialistas seleccionados.

Método estadístico matemático

- **Análisis porcentual:** Este método permitió estudiar los resultados de la validación, no en términos absolutos, sino como porcentaje del activo total.

Para la realización de las encuestas se seleccionó una muestra de 6 especialistas de una población de 28.

Posibles Resultados

- Propuesta de Entorno de Integración Continua para el marco de trabajo jWebSocket.
- Informe detallado con toda la base teórico-práctica sobre la cual se sustenta la solución propuesta.

Para una mejor comprensión de la investigación, el contenido ha sido separado en tres capítulos, aparte de las conclusiones generales, recomendaciones, referencias bibliográficas y bibliografía utilizada, glosario de términos en el cual se detallan los términos técnicos y poco claros utilizados en la elaboración del documento, y los anexos que complementan el trabajo realizado. Los capítulos han sido estructurados de la siguiente manera:

Capítulo 1. Fundamentación Teórica: Se realiza la fundamentación teórica de la investigación. Se expone un estudio del estado del arte sobre los sistemas de Integración Continua en los marcos de trabajo que desarrollan aplicaciones Java en la actualidad, tanto a nivel nacional como internacional.

Capítulo 2. Propuesta de solución: Se describe el proceso de instalación y configuración del entorno de Integración Continua para el marco de trabajo jWebSocket. Se establecen las pautas, reglas y procedimientos a seguir para el uso del entorno de Integración Continua y la realización de las pruebas unitarias.

Capítulo 3. Pruebas y validación de la solución: Se elaboran y documentan las pruebas realizadas a la solución propuesta para demostrar el cumplimiento de los requerimientos de la misma.

CAPITULO 1. FUNDAMENTACIÓN TEÓRICA

Toda investigación científica debe tener una base teórica que sustente y respalde el trabajo a desarrollar. En este capítulo se expondrán los elementos relacionados con entornos de integración continua, se mencionan conceptos fundamentales para la comprensión del objeto de estudio y el campo de acción. Se realizará una caracterización del entorno de integración continua realizado en la Universidad para el Centro de Informatización Universitaria. Finalmente se abordará sobre las tecnologías y herramientas que serán usados para el desarrollo de la solución.

1.1. Conceptos fundamentales

Integración continua

La integración continua es una práctica de desarrollo de software que aparece con el surgimiento de las metodologías ágiles, donde los miembros de un equipo integran su código frecuentemente, normalmente cada persona integra un mínimo de una vez por día. Cada integración genera automáticamente una construcción del proyecto creando así nueva una versión. También ejecuta las pruebas automáticamente, posibilitando la detección de errores lo más pronto posible, esto tributa a una disminución significativamente de problemas de integración, permitiendo al equipo desarrollar software coherente más rápidamente. (Fowler, 2006)

Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día. Todas las pruebas son ejecutadas y tienen que ser aprobadas para que el nuevo código sea incorporado definitivamente. La integración continua a menudo reduce la fragmentación de los esfuerzos de los desarrolladores por falta de comunicación sobre lo que puede ser reutilizado o compartido. Martin Fowler afirma que el desarrollo de un proceso disciplinado y automatizado es esencial para un proyecto controlado, el equipo de desarrollo está más preparado para modificar el código

cuando sea necesario, debido a la confianza en la identificación y corrección de los errores de integración. (Letelier, 2006)

La integración continua tiene como finalidad prevenir errores muy comunes, adelantando todo aquello que pueda ser automatizado en el proceso de construcción de un producto. El objetivo es que con una frecuencia determinada se lleven a cabo esas tareas y en caso de detectar errores se proceda a su resolución cuanto antes, evitando que los errores aparezcan el día de publicación de una nueva versión. La integración continua se basa en un motor de reglas que permite automatizar la ejecución de todas esas tareas comunes y contribuye enormemente a mejorar la calidad del software. (Visión Innovadora de la Calidad del Producto Software, 2009)

Una vez analizado el estudio de los diferentes conceptos de la Integración Continua dados por los autores Martin Fowler, Patricio Letelier y el concepto expuesto en la revista Española de Innovación, Calidad e Ingeniería del Software se asume que la Integración Continua es una práctica de desarrollo de software donde los miembros de un equipo integran su código al menos una vez al día, generando una nueva versión del producto por cada integración, automatizando el proceso de realización de pruebas, previniendo así la detección de errores lo más pronto posible, posibilitando la realización de software de alta calidad en el menos tiempo posible.

Control de Versiones

El control de versiones es una manera de manejar de los cambios en la información. Ha sido durante mucho tiempo una herramienta de suma importancia para los desarrolladores de programas de computadoras, quienes normalmente emplean su tiempo haciendo cambios en sus trabajos y deshaciendo esos cambios al día siguiente. La principal característica de un sistema de control de versiones es brindar la facilidad de manejar el historial de cambios o modificaciones que se han realizado sobre diferentes archivos. La utilidad de un sistema de control de versiones se extiende más allá del límite del mundo de desarrollo de software, ya que estos sistemas no solo se limitan a gestionar archivos de texto, sino que

también gestionan documentos, imágenes, todo tipo de ficheros. Los sistemas de control de versiones guardan toda la información en un repositorio central, accesible a través de la red, brindando facilidades para el trabajo colaborativo entre varios puestos de trabajo y al mismo tiempo proporcionando mayor seguridad y disponibilidad de los datos. (Ben Collins-Sussman, 2004)

El control de versiones no sólo en lo que se refiere a gestionar la vida de un objeto, sino también en lo que tiene que ver con su difusión, es decir: generar diferentes metadatos con distintas cantidades de información sobre un mismo objeto con el fin de distribuirla a un público heterogéneo. (El concepto de metadato. Algo más que descripción de recursos electrónicos, 2003)

Anteriormente fueron analizados los conceptos de Control de Versiones expuestos por los autores Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato y José A. Senso, Antonio de la Rosa Piñero en la revista titulada “El concepto de metadato. Algo más que descripción de recursos electrónicos”. En la presente investigación se asume que Control de Versiones no es más que un sistema capaz de manejar los cambios por los que pasa un fichero determinado desde su creación. Es la herramienta principal para los desarrolladores de software, ya que pueden tener un historial de todas las modificaciones por la que va pasando un proyecto, permitiéndoles regresar a una versión determinada.

Métricas de calidad del código

El principal objetivo de las métricas de calidad del código es obtener un software con la máxima calidad posible. La mayoría de las organizaciones de software más exitosas implementan mediciones del código como parte de sus actividades cotidianas. Para asegurar que producto determinado tenga calidad es necesario realizar un conjunto de medidas con propósitos diferentes que reflejan y describen la conducta del software, estas pueden medir competencia, calidad, desempeño y la complejidad del software, contribuyendo a la mejora de la calidad del producto. (Ludisley la Torre Hernández, 2007)

Se entiende por métricas calidad de código a un conjunto de medidas de software que proporcionan a los programadores una mejor visión del código que están desarrollando. Normalmente, la calidad y la velocidad de desarrollo no van unidas, la perfección tiene en la mayoría de los casos un coste demasiado alto. De esta forma, se tiende a construir software de acuerdo con unos tiempos y unos costes que en muchos casos son tan irreales que hacen que la calidad de los productos obtenidos se vea mermada. (Revisiones de código en el contexto del aseguramiento de calidad. Un caso práctico, 2008)

Anteriormente fueron analizados los conceptos de Métricas de Calidad del Código expuestos por los autores, Ludisley la Torre Hernández, Mariela Cepero Nuñez y el concepto expuesto en la revista Española de Innovación, Calidad e Ingeniería del Software. En la presente investigación se asume que se entiende por Métricas de Calidad de Código un conjunto de medidas tomadas por los desarrolladores para medir la calidad del código creado, también estas proporcionan un conjunto de estadísticas de los códigos realizados como son, cantidad de comentarios, documentación, cantidad de líneas de códigos, entre otra.

Pruebas automatizadas

Uno de los aspectos más importantes de la integración continua es la realización de las pruebas automáticas, su diseño, elaboración y despliegue es una tarea de gran importancia. Esto ayuda a evitar el gran tiempo perdido por los desarrolladores detectando y corrigiendo errores en el código. Gracias a la aparición de la integración continua, aparece la automatización de distintos tipos de pruebas agilizando el ciclo de vida de producción de un software. (Grant, 2010)

Para las pruebas de software se pueden usar diferentes procesos y metodologías, en la mayoría de estas normalmente se realizan las pruebas diariamente. Las pruebas en la integración continua se ejecutan con el objetivo de probar que la evolución del software desarrollado funciona perfectamente, tanto para las funcionalidades que existentes como para las nuevas.

1.2. Estado del Arte

Todo equipo de desarrollo de software debe estar respaldado por un entorno de integración continua que se ajuste a las necesidades y características del tipo de software que realizan. Existen muchas formas de montar un entorno de integración continua en dependencia del objetivo y necesidades del proyecto para el que este diseñado.

En el ámbito nacional, en la Universidad de las Ciencias Informáticas, el Centro de Informatización de la Universidad cuenta con la implantación de un entorno de integración continua, el cual es el trabajo de diploma titulado “Propuesta de entorno de integración continua para el desarrollo de software en el Centro de Informatización de la Universidad”, este fue creado en el año 2010 por el ingeniero Yassef Amed Galarraga Grant. Este entorno se caracteriza por estar compuesto por un servidor de control de versionado de ficheros, el cual es Subversion, como herramienta para la gestión y configuración de proyectos usan Maven, para la realización de las pruebas unitarias usan JUnit y Hudson como servidor de integración continua. (Grant, 2010)

Como se puede ver no tienen una herramienta para la creación de métricas de código. También está diseñado con Hudson como servidor de integración continua, lo cual sería conveniente que fuera cambiado por Jenkins, por problemas de documentación y soporte.

Debido a lo anteriormente expuesto, analizando y caracterizando sus prestaciones, se decide montar un entorno de integración continua para el proyecto jWebSocket. Al realizar un estudio minucioso de las funcionalidades que brinda el entorno estudiado anteriormente, se puede ver que carece de herramientas para medir la calidad del código. Un buen entorno de integración continua debe contar con mecanismos de realización de pruebas automáticas, mecanismos de notificación en caso de encontrar algún error de integración o algún error en el código del proyecto, debe de estar integrado con alguna herramienta que permita medir y crear estadísticas del código en cuanto a cantidad de líneas, complejidad de los

métodos, cantidad de comentarios, todo esto tributa a tener garantizado la realización de un código con la mayor calidad posible.

1.3. Características del entorno de desarrollo del proyecto jWebSocket.

El proyecto jWebSocket está formado por diferentes desarrolladores en diversas partes del mundo. Es un proyecto colaborativo donde se trabaja con muchas herramientas que tributan al desarrollo en línea. El entorno de desarrollo del proyecto jWebSocket se caracteriza por crear soluciones sobre el lenguaje Java, por lo que este cuenta con un conjunto de herramientas las cuales para facilitar el trabajo o desarrollo de estas soluciones.

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 comunicación basado en 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 plugin 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 WebSocket, pero los motores internos se apagan y el anfitrión se utiliza. Esto asegura que no haya mecanismos de seguridad adicionales.

WebSocket 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 WebSocket perfectamente es compatible con SOA (Service Oriented Architectures) 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)

1.4. Tecnologías usadas para el desarrollo de la solución

Marco de Trabajo

Un Marco de Trabajo, más conocido por su significado en inglés como framework, en es un esquema, esqueleto o patrón creado con el objetivo de facilitar el desarrollo y/o la implementación de una aplicación. (¿Qué es un 'framework'?, 2006) Un marco de trabajo se puede considerar como un conjunto de librerías y funcionalidades genérica incompleta y configurable a la que podemos añadirle las últimas piezas para facilitar el proceso de construcción de una aplicación, 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 objetivo de facilitar el desarrollo de software, permitiendo a los desarrolladores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel para proveer un sistema funcional.

Marco de trabajo para realizar pruebas unitarias

A continuación se expondrán los principales marcos de trabajos más utilizados para la realización de las pruebas unitarias. Es el caso de JUnit para proyectos desarrollados con Java y NUnit para aplicaciones desarrolladas con .NET.

NUnit

NUnit es una de esas herramientas utilizada para escribir y ejecutar pruebas en .NET, es un framework desarrollado en C# que ofrece las funcionalidades necesarias para implementar pruebas en un proyecto. Además provee una interfaz gráfica para ejecutar y administrar las mismas. También se utiliza para realizar pruebas unitarias a .NET⁶.

Lo que hace tan efectiva la TDD (Test Driven Development) es la automatización de las pruebas de programador, (programmer unit tests). El hecho de utilizar TDD implica 3 acciones: escribir las pruebas, escribir el código que debe pasar las pruebas y refactorizar para eliminar el código duplicado. La primera se lleva a cabo de una manera sencilla y simple utilizando NUnit, esta soporta los lenguajes bases de .NET como C#, J#, VB y C++.

NUnit es una herramienta que se encarga de analizar ensamblados generados por .NET, interpretar las pruebas inmersas en ellos y ejecutarlas. Utiliza atributos personalizados para interpretar las pruebas y provee además métodos para implementarlas. En general, NUnit compara valores esperados y valores generados, si estos son diferentes la prueba no pasa, en caso contrario la prueba es exitosa.

NUnit carga en su entorno un ensamblado y cada vez que lo ejecuta, o mejor, ejecuta las pruebas que contiene, lo recarga. Esto es útil porque se pueden tener ciclos de codificación y ejecución de pruebas simultáneamente, así cada vez que se compile no tiene que volver a cargar el ensamblado al entorno de NUnit si no

⁶ .NET: Marco de trabajo creado por la empresa Microsoft.

que este siempre obtiene la última versión del mismo. NUnit ofrece una interfaz simple que informa si una prueba falló, pasó o fue ignorada.

NUnit basa su funcionamiento en dos aspectos, el primero es la utilización de atributos personalizados. Estos atributos le indican al framework de NUnit que debe hacer con determinado método o clase, es decir, estos atributos le indican a NUnit como interpretar y ejecutar las pruebas implementadas en el método o clase. El segundo son las aserciones, que no son más que métodos del framework de NUnit utilizados para comprobar y comparar valores. (Guilarte, 2007)

jUnit

jUnit es un conjunto de librerías de licencia libre, creadas con el objetivo de realizar pruebas unitarias a proyectos desarrollados en Java. Permite realizar una ejecución de las clases Java de forma controlada para así evaluar cada método de estas verificando si el comportamiento de este es el esperado. Es decir, en dependencia del valor de entrada se espera un valor de salida, si el método cumple con lo esperado, entonces jUnit dice que el método pasó la fase de prueba exitosamente.

Mediante jUnit también se pueden controlar las pruebas de regresión. Esto es necesario cuando una parte del código ha sido modificado y se desea comprobar que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación. (Guilarte, 2007) (JUnit, 2012)

Anteriormente se analizaron dos de los principales marcos de trabajo para la realización de las pruebas unitarias en proyectos desarrollados con Java y lenguajes de la plataforma .NET. Se decide tomar el marco de trabajo jUnit para la realización de las pruebas unitarias en el entorno de trabajo del proyecto jWebSocket, debido a que este está diseñado para proyectos realizados con el lenguaje Java y mientras que NUnit es para aplicaciones creadas con la plataforma .NET.

Herramientas de control de versiones

CVS

El Sistema de Control de Versiones más conocido por sus siglas en inglés CVS (Concurrent Versions System) es una herramienta que mantiene el registro de todo el trabajo y los cambios en la implementación de un proyecto informático (de software) y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren en el mismo.

CVS es difundido bajo la Licencia Pública General de GNU (licencia GPL) más conocida por sus siglas en inglés: GNU General Public License. El hecho de ser libre, unido a sus características avanzadas, le dieron gran popularidad en la comunidad del software libre. Fue desarrollado por GNU (GNU's Not Unix), cuyo sitio lo distribuye, denominándolo "paquete GNU" con aplicaciones básicas a través de esta página.

Esta herramienta tiene una arquitectura cliente-servidor. El servidor de CVS en Linux puede ser usado tanto por clientes en Linux como en Windows, e incluso en otros sistemas operativos usados a nivel mundial. Es el encargado de almacenar todos los archivos del proyecto, así como su historial de cambios. Mientras que los clientes se conectan al servidor para obtener una copia íntegra de los archivos correspondientes a un módulo determinado. (Bisset, 2009)

Microsoft Visual SourceSafe (VSS)

Microsoft Visual SourceSafe conocida como VSS es una herramienta de control de versiones a nivel de archivos que forma parte de Microsoft Visual Studio. Permite que varias organizaciones trabajen en distintas versiones del proyecto al mismo tiempo y es importante en el proceso de desarrollo de software ya que se usa para mantener versiones de código paralelas. Ayuda al equipo a evitar la pérdida por accidente de archivos. Es un sistema de control de versiones bajo una licencia de tipo Propietaria. Permite realizar un seguimiento de las versiones anteriores de un

archivo. Admite la bifurcación, el uso compartido, la combinación y la administración de versiones de archivos. (Grant, 2010)

Subversion

Subversion es un software para el control de versiones de archivos. Este es uno de los más usados entre los desarrolladores de software, ya que es distribuido bajo licencia libre de tipo Apache, es multiplataforma. Fue creado con el objetivo principal de remplazar al popular CVS⁷. (Bisset, 2009) (Apache, 2012)

Ventajas de Subversion:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- La creación de ramas y etiquetas es una operación más eficiente. Tiene costo de complejidad constante ($O(1)$) y no lineal ($O(n)$) como en CVS.
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Puede ser servido mediante Apache, sobre WebDAV/DeltaV. Esto permite que clientes WebDAV utilicen Subversion de forma transparente.
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).
- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.
- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autenticar archivos (SQL, LDAP, PAM, etc.).

Carencias:

- El manejo de cambio de nombres de archivos no es completo. Lo maneja como la suma de una operación de copia y una de borrado.

⁷ **CVS:** Concurrent Versions System de sus siglas en inglés, es un sistema de control de versiones creado por GNU.

- No resuelve el problema de aplicar repetidamente parches entre ramas, no facilita llevar la cuenta de qué cambios se han realizado. Esto se resuelve siendo cuidadoso con los mensajes de commit.

Anteriormente se analizaron una de las dos principales herramientas de control de versiones. Para gestionar el versionado de los ficheros en el entorno de trabajo del proyecto jWebSocket se decide tomar Subversion, debido a que es distribuido bajo licencia libre de tipo Apache, es multiplataforma y fue creado con el objetivo principal de remplazar al popular CVS agregándole nuevas funcionalidades que este no tenía.

Herramientas de gestión y configuración de proyectos

Apache Ant

Apache Ant es una herramienta gratuita de construcción de dominio público utilizada en la compilación y creación de programas Java. Se basa en la ejecución de tareas que se modelan en un fichero el XML. Es una de las herramientas J2EE⁸ de construcción más usada. Es multiplataforma, ya que está escrito en Java. Licencia Apache 2.0. Utiliza ficheros de construcción escritos en XML. Puede extenderse fácilmente creando nuevas tareas. (Grant, 2010)

Maven

Maven es una herramienta para la gestión y comprensión de proyectos Java. Es capaz de encargarse de la gestión de dependencias, construcción de los artefactos, generación de la documentación, etc. Su configuración es más simple, reutilizable y consistente que la de Ant. Una vez instalado y configurado, no es necesario mantenerlo. Es rápido el acceso a los paquetes ya descargados. Cuenta con una licencia libre, la cual es: Licencia Apache 2.0. Es independiente de una conexión a internet, excepto cuando se necesita algún paquete nuevo. (Grant, 2010)

Anteriormente se caracterizaron una de las dos principales herramientas de gestión y configuración de proyectos desarrollados en Java. Se decide tomar Maven como

⁸ **J2EE**: Edición empresarial de la plataforma de Java (Java Platform, Enterprise Edition o Java EE)

herramienta de gestión y configuración de proyectos para el marco de trabajo de `WebSocket`, debido a que es capaz de encargarse de la gestión de dependencias, construcción de los artefactos, generación de la documentación, no es necesario mantenerlo y es independiente de una conexión a internet de no ser necesario algún paquete nuevo.

Herramienta de gestión de repositorios de artefactos de construcción

Sonatype Nexus

Sonatype Nexus es una herramienta de gestión de repositorios de artefactos de construcción. Nexus maneja los artefactos de software requeridos para el desarrollo, despliegue y aprovisionamiento de una aplicación. Nexus simplifica el mantenimiento de tus propios repositorios internos y el acceso a los repositorios externos. Con Nexus se puede controlar completamente el acceso y desarrollo de cada artefacto en una organización desde una localización determinada.

Este cuenta con dos versiones con licencias distintas. Nexus Open Source, provee un nivel esencial de control sobre los repositorios externos de Maven que se utilizan y los repositorios internos que se crean. Provee una infraestructura y servicios para la organización que utilicen administradores de repositorios para obtener y entregar software. La otra versión es Nexus Professional, este fue concebido para responder a las necesidades de una empresa. Es un punto central de acceso a repositorios externos que provee los controles necesarios para asegurar que solo los artefactos aprobados entren en el entorno de desarrollo de software. (Sonatype, 2012)

Apache Archiva

Apache Archiva es un programa de código abierto para el manejo de repositorios desarrollado por Apache Software Foundation. Forma parte de los llamados "Proyectos Top Level" de la Fundación Apache. (Archiva, 2012)

Este software ofrece funciones para la mantención (proxying de repositorios remotos, seguridad de acceso, almacenamiento de artefactos, suministro, revisión,

indexación, estadísticas de uso, escaneo) de repositorios, siendo capaz de interactuar con herramientas como Maven, Continuum y Continuum Ant, destinadas a producir una aplicación lista para usar.

Apache Archiva cuenta con diversas características lo que lo hace muy funcional como son:

- Información del proyecto, donde podemos encontrar los datos principales de los creadores de un artefacto o jar.
- Cuenta con un sistema avanzado para la búsqueda de artefactos.
- Cuenta con un mecanismo de cacheo para artefactos en línea.
- Permite agregar nuevos artefactos.
- Cuenta con una administración grafica web.

Anteriormente se caracterizaron dos potentes herramientas de gestión de repositorios de artefactos de construcción para proyectos desarrollados en Java. Se selecciona la herramienta Apache Archiva para su uso en el marco de trabajo del proyecto jWebSocket primeramente porque es un software de uso libre, también cuenta con un sistema avanzado para la búsqueda de artefactos y tiene una administración grafica web lo que facilita mucho el trabajo con este servidor.

Servidores de integración continua

Hudson

Hudson es un servidor de Integración Continua para la construcción de proyectos en Java. Es un producto código abierto distribuido bajo licencia MIT. Hudson trabaja con herramienta de control de versiones como CVS, Subversion, Git y Clearcase y puede ejecutar proyectos basados en Apache Ant y Maven. (Hudson, 2012)

Durante noviembre de 2010, una incidencia surgió en la comunidad de Hudson con respecto a la infraestructura usada, la cual creció hasta cuestionar el control de Oracle sobre el proyecto. A pesar del acuerdo entre los principales contribuidores al proyecto y Oracle en muchos aspectos, el uso de la marca "Hudson" no fue cedido

por Oracle, por lo que en enero de 2011 se propuso cambiar el nombre de "Hudson" por "Jenkins". La propuesta fue apoyada mayoritariamente creándose el proyecto. (Smart, 2011)

Apache Continuum

Continuum es un servidor de Integración Continua para la construcción de proyectos en Java. Es un producto de código abierto y distribuido bajo licencia Apache 2. Para la construcción de sus proyectos brinda soporte para el uso de Ant, Maven 1 y 2 y Shell Scripts como herramientas de automatización. Brinda soporte para los sistemas de control de versiones: CVS, Subversion, Clearcase, Perforce. En su versión 1.0.3 brinda soporte para el envío de correos electrónicos así como a cuatro mecanismos de notificación, tres de ellos son protocolos de mensajería. (Continuum, 2012)

Jenkins

Jenkins es un servidor de Integración Continua hecho en Java. Está basado en el proyecto Hudson. Actualmente cuenta con el soporte de toda la antigua comunidad de Hudson, la cual se encuentra ahora desarrollando para Jenkins. Soporta herramientas de control de versiones como CVS, Subversion, Git, Mercurial, Perforce y Clearcase y puede ejecutar proyectos basados en Apache Ant y Maven. Esta liberado bajo licencia MIT. (Smart, 2011)

Jenkins ha dominado una amplia parte del mercado de las herramientas de integración continua. Es utilizado por equipos que desarrollan con diferentes tecnologías y lenguajes como .NET, Ruby, Groovy, Grails, PHP y más, así como Java. (Smart, 2011)

Jenkins es uno de los servidores de integración continua más fáciles de usar. Su interfaz de usuario es simple y sencilla. Cuenta con una gran comunidad que no para de brindar soporte y mantener una actualizada documentación de este servidor, debido a esto se decide utilizar Jenkins como servidor de Integración Continua para el marco de trabajo de jWebSocket.

Herramienta para crear métricas de código

Understand

Understand es una herramienta privativa que permite mantener, medir y crear análisis del código fuente de un programa. Tiene soporte para varios lenguajes como Ada, C/C++, C#, Fortran, Java, Pascal, Cobol, Python. También brinda soporte para lenguajes web como PHP, HTML, CSS y Javascript, de los cuales permite obtener información como total de líneas, líneas en blanco, comentarios, entre otros.

Sonar

Sonar es una excelente herramienta para obtener métricas de calidad del código de proyectos escritos en Java. Está distribuido bajo una licencia libre, licencia LGPL. Como su propia web dice, Sonar es una plataforma que permite gestionar la calidad del código controlando los siete ejes principales de dicha calidad de código, como son:

- Arquitectura y Diseño.
- Duplicaciones de código.
- Pruebas unitarias.
- Complejidad.
- Errores potenciales.
- Reglas de codificación.
- Comentarios.

Tiene una amplia cobertura para diferentes lenguajes de programación como Java, C, C#, Flex, Natural, PHP, PL/SQL, Cobol y Visual Basic 6. Cuenta con una simple interfaz de usuario mediante la cual brinda información a los desarrolladores como errores potenciales en el código, escasez de comentarios, clases demasiadas complejas, escasez de cobertura en las pruebas unitarias, y más, permitiendo obtener métricas y estadísticas del código. (Sonar, 2012)

Se decide seleccionar Sonar como herramienta para la creación de métricas y estadísticas en los proyectos desarrollados con el marco de trabajo jWebSocket debido a que este es software libre, también porque este tiene una gran cantidad de complementos que permite una buena integración con el servidor de integración continua Jenkins brindando más funcionalidades para la solución.

Conclusiones del capítulo

En este capítulo se destacaron los elementos relacionados con entornos de integración continua. Se mencionaron conceptos fundamentales para la comprensión del objeto de estudio y el campo de acción, como es el concepto de integración continua, que es un control de versiones, se abordaron temas relacionados con las métricas de calidad del código y pruebas automatizadas. Se realizó una caracterización del entorno de integración continua realizado en la uci para el Centro de Informatización Universitaria. Al final de este capítulo se hizo una caracterización detallada y selección las tecnologías y herramientas que serán usadas para conformar el entorno de integración continua para proyectos desarrollados con el marco de trabajo de jWebSocket.

CAPITULO 2. PROPUESTA DE UN ENTORNO DE INTEGRACIÓN CONTINUA.

Para un grupo de desarrollo de software la calidad de su producto debe ser uno de los principales aspectos a tener en cuenta. En la actualidad la Integración Continua es uno de los principales recursos para lograr un producto de alta calidad. A continuación se dará a conocer el estado actual del proceso de desarrollo en proyectos desarrollados con el marco de trabajo jWebSocket. Posteriormente se realizará un análisis a la encuesta desarrollada para diagnosticar el problema. Después se expondrá un resumen de la propuesta de solución, dando a conocer para qué usuarios está enfocada la solución propuesta. Más adelante se dará a conocer el principio de funcionamiento del entorno de integración continua propuesto. Finalmente se expondrá el proceso de instalación y configuración de las herramientas seleccionadas para la conformación del entorno de integración continua para proyectos desarrollados con el marco de trabajo de jWebSocket.

2.1. Estado actual del proceso de desarrollo de proyectos con el entorno de jWebSocket

El proyecto jWebSocket está formado por diversos desarrolladores de distintas partes del mundo. En la actualidad la integración de códigos de cada uno de estos desarrolladores es realizada de forma manual por el líder del proyecto existiendo grandes problemas en el proceso de integración.

Realizar los procesos de integración manualmente en un grupo de desarrollo de software trae como consecuencia un gran esfuerzo por parte del miembro del proyecto encargado de realizar esta tarea. Existe también poca agregación de archivos al repositorio, lo que atenta también en el proceso integración y consigo la existencia de poca generación de versiones. Las pruebas son realizadas manualmente y solo al final de cada integración, lo que trae como consecuencia muchos errores en los códigos, perdiendo más tiempo y esfuerzo solucionando estos problemas los cuales solo son encontrados al final de cada integración.

El entorno de trabajo del grupo jWebSocket tiene los problemas anteriormente descritos. A continuación se expone una imagen donde se describe el entorno de trabajo del grupo jWebSocket con los problemas anteriormente mencionados.



Figura 1.1 Entorno de trabajo actual del proyecto jWebSocket. Fuente: Elaboración propia.

2.2. Resultados de la encuesta realizada a especialistas de diferentes proyectos de la facultad para la determinación del diagnóstico del problema

Para lograr un mejor diagnóstico del problema se procede a realizar una encuesta que cuenta con cinco preguntas principales, enfocadas a los factores que atentan contra la calidad de la realización de un proyecto. A continuación se mostrara las preguntas realizadas en la encuesta, las cuales tiene valores de respuestas de desde 1 hasta 5 los respondiendo a muy alta, alta, media, baja y muy baja respectivamente.

Listado de las preguntas realizadas en la encuesta:

1. ¿Cómo valora los problemas de integración de cada uno de los códigos de los diferentes desarrolladores?
2. ¿Considera usted que exista una seguida generación de versiones del proyecto desarrollado?
3. ¿Considera usted que existe una seguida realización de pruebas al código?
4. ¿Considera usted que existe una pérdida de tiempo por parte de los desarrolladores arreglando defectos en el código?
5. ¿Considera usted que existan problemas de entrega en tiempo y finalización de los proyectos?

A continuación se muestra una tabla con los resultados de las preguntas para la determinación del diagnóstico del problema:

	Especialista 1	Especialista 2	Especialista 3	Especialista 4	Especialista 5	Especialista 6
Pregunta 1	2	1	1	1	2	1
Pregunta 2	5	5	5	4	5	5
Pregunta 3	4	5	5	5	5	5
Pregunta 4	1	1	1	1	2	1
Pregunta 5	2	2	2	3	2	1

Tabla 1 Resultados de la encuesta para la determinación del diagnóstico del problema.

Como se puede observar por los resultados de la tabla anteriormente expuesta, en los proyectos de la facultad que usan el marco de trabajo de jWebSocket presentan graves problemas en cuanto a la integración de los códigos de cada uno de los desarrolladores. También existe una escasa generación de versiones de los productos realizados y se realizan pocas pruebas al código a medida que se va desarrollando. También existe una alta pérdida de tiempo por parte de los desarrolladores corrigiendo problemas en el código que no son detectados inmediatamente. Todos estos parámetros atentan contra la entrega en tiempo de un producto determinado.

2.3. Propuesta de solución

La propuesta de solución está diseñada para proyectos que usen el marco de trabajo de JWebSocket, por lo que está concebida para realizar la integración y pruebas automáticas sobre códigos hechos en Java. Esta propuesta será usada mayormente en las etapas de desarrollo y pruebas de un proyecto de desarrollo de software sin importar la metodología que este utilice.

Para dar solución a la problemática anteriormente expuesta se propone crear la siguiente propuesta de entorno de integración continua la cual está compuesto por Jenkins como servidor de integraciones y pruebas automatizadas, como marco de trabajo para realizar pruebas unitarias se usara jUnit, la herramienta de control de versiones que se usara es Subversion, Maven como herramienta de gestión y configuración de proyectos, para la gestión de repositorios de artefactos de construcción se usara Apache Archiva y el servidor de Sonar para la creación de métricas de calidad del código.

2.4. Funcionamiento del entorno de integración continua propuesto

El entorno de integración continua propuesto esta diseñado con el objetivo de que funcione completamente de forma automática, para lograr esto se realizara una configuración en los hooks o ganchos del servidor Subversion, de manera que cuando un desarrollador realice un commit este ejecute de forma remota el proyecto en el servidor de integración continua Jenkins asociado al commit realizado, generando una compilación del proyecto completo y una integración automática de todos los códigos de los distintos desarrolladores.

En caso de ser encontrado algún error en el código subido o de fallar una prueba unitaria, será notificado por correo al usuario que provocó el fallo y también a los usuarios registrados en la lista de distribución referente al proyecto en el cual fue detectado el error. Cuando se realice otro commit en el servidor de Subversion referente al proyecto donde se detectó el error anterior y este haya sido corregido, el servidor de integración continua cataloga ese suceso como importante y procede

a realizar otra notificación al usuario que lo corrigió y a la lista de distribución del proyecto.

También con cada integración es generada una nueva versión del proyecto y se realiza una ejecución remota al servidor de métricas Sonar, el cual genera un conjunto de métricas asociadas al proyecto compilado como son: cantidad de clases, métodos y paquetes, complejidad de los códigos, cantidad de comentarios de documentación y otros comentarios, cantidad de errores de codificación que atentan contra la calidad del producto. También es generado un documento PDF con un resumen de todas las métricas realizadas al proyecto.

Como se puede apreciar este entorno de integración continua está diseñado para su completo funcionamiento sin necesidad de una persona que lo atienda. Todos los procesos por los que debe pasar un proyecto fueron automatizados, se comienza por la realización de un commit, después se realiza la compilación del proyecto completo, las pruebas unitarias son ejecutadas automáticamente, se notifica por correo a los usuarios necesarios en caso de ocurrir algún problema y finalmente son realizadas las métricas del proyecto obteniendo un documento con la descripción de estas.

2.5. Proceso de descarga de la aplicación

Para la instalación y configuración del entorno de integración continua diseñado para los proyectos desarrollados con el marco de trabajo jWebSocket es necesario descargar la última versión estable de los siguientes programas:

- Apache Archiva (<http://archiva.apache.org/>)
- Jenkins (<http://jenkins-ci.org/>)
- Sonar (<http://www.sonarsource.org/>)

2.6. Características del entorno de instalación

Los requisitos mínimos para el buen funcionamiento del entorno de integración son; una computadora personal con Ubuntu Server preferentemente la versión 11.10 de 64 bits con los siguientes paquetes instalados:

- Servidor Apache2
- Servidor de base de datos MySQL
- Servidor de Apache Tomcat6
- Máquina virtual openjdk-7

2.7. Proceso de instalación

Instalación de Subversion

A continuación se mostrará como instalar el Subversion juntos con su librería para apache y su paquete de herramientas, para esto es necesario poner el siguiente comando en la consola:

```
sudo apt-get install subversion libapache2-svn subversion-tools apache2-mpm-prefork
```

Instalación de Apache Archiva

Para la instalación de Apache Archiva primeramente es necesario crear el directorio ARCHIVA_HOME el cual contendrá la instalación. Este directorio se creará preferentemente dentro del home de la sesión iniciada.

```
mkdir /home/user/ARCHIVA_HOME
```

Después se procede a descomprimir en este directorio la última versión descargada del Apache Archiva.

Finalmente se procede a iniciar Archiva, para esto es necesario posicionarse en el directorio bin que se encuentra dentro de ARCHIVA_HOME y ejecutar el siguiente comando:

```
./archiva start
```

Instalación de Maven2

Para la instalación de Maven2 solamente es necesario ejecutar el siguiente comando en la consola con privilegios de administración:

```
apt-get install maven2
```

Instalación de Jenkins

Para la correcta instalación de Jenkins primeramente es necesario instalar una dependencia, el paquete “daemon”, para lo cual es necesario ejecutar el siguiente comando en la consola con privilegios de administración:

```
apt-get install daemon
```

Una vez instalado la dependencia “daemon” se puede comenzar con la instalación de Jenkins. Para esto es necesario ubicarse en el directorio donde se encuentra descargado el paquete de la última versión de Jenkins, en este caso se instalara la versión 1.448, para eso es necesario ejecutar el siguiente comando en la consola con privilegios de administración:

```
dpkg -i jenkins_1.448_all.deb
```

Instalación de Sonar

Primeramente es necesario instalar el servidor de base de datos MySQL, para esto es necesario ejecutar el siguiente comando en la consola con privilegios de administración:

```
Apt-get install mysql-server
```

Posteriormente es necesario crear la base de datos para el servidor de sonar, para esto se ejecuta el siguiente comando en la consola:

```
mysqladmin -p create sonar
```

Una vez creada la base de datos se puede comenzar con la instalación de sonar. Lo primero es descargar la última versión estable de esta herramienta y descomprimirla en algún lugar del disco duro el cual será reconocido como SONAR_HOME.

Posteriormente es necesario abrir el fichero SONAR_HOME/conf/sonar.properties para establecer las principales configuraciones como conexión a la base de datos sonar. El archivo es bastante auto explicativo, se comentan las tres líneas en las cuales se especifica el uso de la base de datos embebida:

#DATABASE

#Comment the 3 following lines to deactivate the default embedded database (used only for tests and demos)

```
#sonar.jdbc.url:jdbc:derby://localhost:1527/sonar;create=true
```

```
#sonar.jdbc.driverClassName:org.apache.derby.jdbc.ClientDriver
```

```
#sonar.jdbc.validationQuery:values(1)
```

Y descomentar las entradas que se refieren al uso de una base de datos MySQL:

#MySQL

#uncomment the 3 following lines to use MySQL

```
sonar.jdbc.url: jdbc:mysql://localhost:3306/sonar?useUnicode=true&characterEncoding=utf8
```

```
sonar.jdbc.driverClassName: com.mysql.jdbc.Driver
```

```
sonar.jdbc.validationQuery: select 1
```

Posteriormente se establecen los datos de configuración para la conexión con la base de datos creada anteriormente con nombre sonar. Esto se hace en la sesión del fichero de configuración nombrada “generic settings”:

generic settings

```
sonar.jdbc.username: root
```

```
sonar.jdbc.password: mipassw
```

Una vez establecidos los parámetros de configuración necesarios, se procede a generar el .war el cual se va a desplegar en una instancia de Tomcat. Para esto es necesario situarse en el directorio SONAR_HOME/war y ejecutar el siguiente comando en la consola:

```
./build-war.sh
```

Esto genera en el directorio un fichero llamado sonar.war listo para desplegarse en cualquier instancia de Tomcat. Ahora basta con copiar este fichero en el directorio /var/lib/tomcat6/webapps y arrancar el servidor Tomcat para que sea desplegada la aplicación. Posteriormente se puede acceder a sonar mediante la URL http://ip_del_servidor:8080/sonar.

2.8. Opciones de configuración

Configuración del servidor de Subversion

Primeramente es necesario crear un directorio donde el servidor de Subversion guardará el repositorio. A continuación se mostrará el comando para crear este directorio dentro del directorio home del usuario actual:

```
mkdir /home/user/svn
```

Posteriormente se creará el repositorio dentro del directorio creado, para eso es necesario ejecutar el siguiente comando en la consola:

```
svnadmin create /home/user/svn/jwebsocket
```

Después se creara la estructura de carpetas del repositorio, branches, tags y trunk.

```
svn mkdir --message="Setting up the directories..."
```

```
file:///home/user/svn/jwebsocket/trunk
```

```
file:///home/user/svn/jwebsocket/tags
```

```
file:///home/user/svn/jwebsocket/branches
```

El siguiente paso es poner como propietario a www-data del directorio que contiene el repositorio.

```
chown www-data:www-data /home/carlos/svn/jwebsocket/ -R
```

Después movemos el fichero de configuración principal del repositorio de /home/user/svn/jwebsocket.conf/authz para la carpeta raíz del repositorio /home/user/svn/authz y se establece la siguiente configuración en el fichero:

```
/etc/apache2/mods-available/dav_svn.conf
```

```
<Location /svn>
```

```
#Fijamos la ruta base del repositorio
```

```
DAV svn
```

```
#Y este es el camino donde estarán todos los repos
```

```
SVNParentPath /home/carlos/svn/
```

```
#modo de autenticación
```

```
AuthType Basic
```

```
#Nombre del repo
```

```
AuthName "Subversion Repository jWebSocket"
```

```
#aquí nos detendremos porque esta línea puede o no comentarse, solo deberá usarse cuando #solo
se está autenticándose por los usuarios c$
AuthUserFile /etc/apache2/passwords
# AuthBasicProvider Idap
# AuthzLDAPAuthoritative on
# AuthLDAPURL "ldap://10.208.0.3:389/OU=Personas, DC=hab, DC=uci,
DC=cu?uid?sub?(objectClass=*)"
# AuthLDAPURL "ldap://10.0.0.3:389/DC=uci, DC=cu?uid?sub?(objectClass=*)"
Require valid-user
# Activamos la autorización para los permisos vía mod_authz_svn
AuthzSVNAccessFile /home/user/svn/authz
</Location>
```

Ahora se puede acceder al repositorio desde la siguiente dirección:
http://ip_del_servidor/svn/jwebsocket/.

Por último es necesario establecer una configuración en los ganchos del servidor de Subversion para lograr que este avise al servidor de integración continua cuando se realice algún cambio en su estructura de carpeta y este prosiga a realizar la integración de los nuevos cambios de manera automática.

Primeramente es necesario modificar el fichero
 /home/usuario/svn/jwebsocket/hooks/post-commit.tmpl y agregar esta línea al final del este:

```
# como último parámetro se le debe pasar los directorio del repositorio para los cuales se quiere
crear la alerta.
/home/usuario/svn/jwebsocket/jenkins-launch-build.sh $REPOS $REV nombreDelDirectorioSVN
nombreDelProyectoJenkins
```

Posteriormente se prosigue a crear el fichero
 /home/usuario/svn/jwebsocket/jenkins-launch-build.sh y configurarlo como se muestra a continuación:

```
#!/bin/bash
# Este script comprueba si se han hecho cambios en un directorio concreto,
# y en tal caso lanza una build en Hudson
```



```

REPOS="$1"
REV="$2"
PROJECT_NAME="$3"
JENKINS_JOB="$4"

JENKINS_USER=admin
JENKINS_PASSWORD=mipassw
JENKINS_HOST=10.208.7.201:8002
IS_PROJECT_CHANGED=`svnlook dirs-changed $REPOS --revision $REV | fgrep
$PROJECT_NAME`
if [[ -n $IS_PROJECT_CHANGED ]]; then
    wget --quiet --auth-no-challenge --no-check-certificate --http-user=$JENKINS_USER --http-
password=$JENKINS_PASSWORD
http://$JENKINS_HOST/job/$JENKINS_JOB/build?token=TOKEN
    exit 0
fi

```

Configuración de Apache Archiva

El servidor de archiva una vez instalado este funciona por defecto por el puerto 8080, el mismo puerto que usa Tomcat, por lo que se hace necesario cambiar el puerto de este a 8888. Para cambiar el puerto de Archiva es necesario ir a fichero de configuración ARCHIVA_HOME/conf/jetty.xml, línea 66 y cambiar el puerto por defecto de 8080 a 8888, a continuación se mostrará como debe quedar esta configuración.

```
<Set name="port"><SystemProperty name="jetty.port" default="8888"/></Set>
```

Configuración de Maven2

Para configurar maven2 es necesario modificar el fichero /etc/maven2/settings.xml y agregar los mirrors. A continuación se muestra un ejemplo de cómo debe quedar este fichero:

```

<mirror>
  <id>archiva.default</id>
  <url>http://10.208.7.201:8888/archiva/repository/internal/</url>

```

```

    <mirrorOf>*</mirrorOf>
</mirror>
<mirror>
    <id>archiva.apache.snapshots</id>
    <url>http://10.208.7.201:8888/archiva/repository/snapshots/</url>
    <mirrorOf>apache.snapshots</mirrorOf>
</mirror>

```

Configuración de Jenkins

Jenkins al instalarse usa el puerto 8080 por defecto, se hace necesario cambiarlo debido a que es el mismo puerto que usa Tomcat. A continuación se cambiará la configuración de Jenkins para que corra por el puerto 8002. Para cambiar el puerto por defecto de Jenkins se procede a editar el archivo de configuración `/etc/default/jenkins` donde se cambia el siguiente parámetro de configuración:

```

# port for HTTP connector (default 8080; disable with -1)
HTTP_PORT=8002

```

Seguidamente se reinicia el servicio de Jenkins:

```

/etc/init.d/jenkins restart

```

Ahora se puede acceder al servidor de Jenkins mediante la siguiente dirección: `http://ip_del_servidor:8002/`.

Seguidamente se procede a activar los disparadores de cada uno de los proyectos creados en Jenkins para que el servidor de Subversion sea capaz de avisarle cuando ocurra una modificación y el servidor de integración continua realice automáticamente la integración de los códigos. Para esto es necesario ir a la configuración de cada proyecto creado en Jenkins, a la sesión de disparadores de ejecución y establecer la configuración que se muestra a continuación:

Disparadores de ejecuciones

- ☒ Ejecutar siempre que cualquier 'SNAPSHOT' de los que dependa sea creado
- ☐ Ejecutar después de que otros proyectos se hayan ejecutado
- ☒ Lanzar ejecuciones remotas (ejem: desde 'scripts')

Identificador de seguridad

Usar esta dirección web para lanzar la ejecución remota `JENKINS_URL/job/testingMaven/build?token=TOKEN_NAME` o `/buildWithParameters?token=TOKEN_NAME`
 Añadir el parámetro `&cause=Cause+Text` para componer el texto que será incluido en el texto describiendo la causa de la ejecución

- ☐ Consultar repositorio (SCM)
- ☐ Ejecutar periódicamente

Figura 1.2 Ejemplo de configuración de los disparadores de ejecuciones en Jenkins. Fuente:
Elaboración propia.

Finalmente se procede a configurar el plug-in para integrar Jenkins con Sonar. Para comenzar es necesario configurar el modo de acceso a internet, en este caso se usará una conexión mediante un servidor proxy, para esto es necesario ir al menú de administración de jenkins, administrar plug-ins, Configuración avanzada. A continuación se mostrará un ejemplo de configuración del proxy.

Actualizaciones disponibles Todos los plugins Plugins instalados **Configuración avanzada**

Configuración de proxy

Servidor

Puerto

Usuario

Contraseña

Figura 1.3 Ejemplo de configuración del proxy para la instalación de plug-ins en Jenkins. Fuente:
Elaboración propia.

Finalmente se procede a instalar el plug-in para Sonar. Para esto es necesario ir al menú de administración de Jenkins, administrar plug-ins, en la pestaña todos los plug-ins, seleccionar el plug-in “sonar plug-in” y presionar el botón “download now and install after restart”.

2.9. Administración de la aplicación

Administración de Subversion

El primer aspecto en la administración de un repositorio es la creación de los usuarios. Para esto se hace uso del comando `htpasswd`, por primera vez esto se hace usando el parámetro `-mc`, a partir de ese momento se cambia el parámetro `-mc` por solamente `-c`. A continuación se expone el ejemplo de la creación del primer usuario para el Subversion:

```
htpasswd -mc /etc/apache2/passwords usuario
```

El otro pasó es la asignación de permisos a los diferentes directorios del repositorio. Para esto es necesario editar el fichero de configuración `/home/usuario/svn/authz`. Lo primero que se debe hacer es asignar grupos de usuarios, esto se hace en la cápsula `[groups]`, seguidamente se pone la relación de todos los usuarios que componen el grupo separados por coma. A continuación se expone un ejemplo de la creación de un grupo y la asignación de varios usuarios a este.

```
[groups]
```

```
admins = carlosfeyt, vbarzana
```

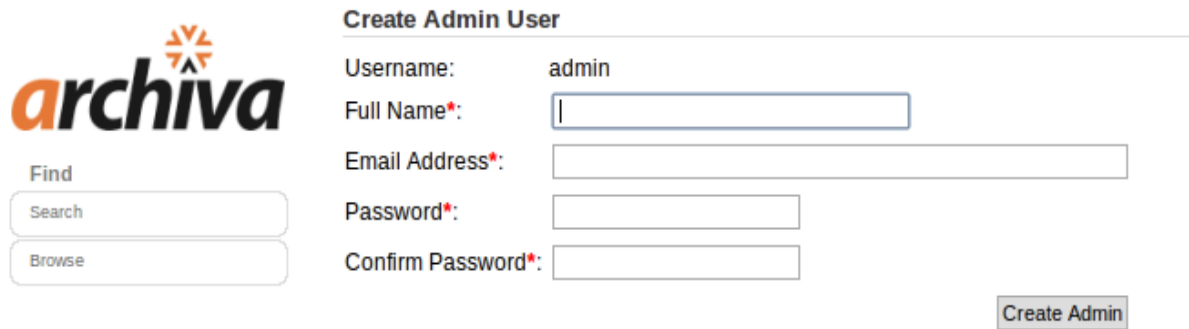
El siguiente paso es como asignar permisos a un directorio específico para un grupo determinado. Esto se hace encerrando la ruta del directorio entre corchetes, posteriormente se pone el grupo al cual se le va a asignar el permiso empezando con una arroba o se puede poner un asterisco para especificar todos los grupos creados anteriormente, después se iguala a los permisos que va a tener sobre el directorio seleccionado. Los permisos pueden ser de lectura y escritura. A continuación se expone un ejemplo de la asignación de permisos de escritura y lectura a al directorio raíz del repositorio al grupo `admins` creado anteriormente:

```
[/]
```

```
@admins = rw
```

Administración de Apache Archiva

Cuando se accede a Apache Archiva por primera vez este exige la creación de un usuario de administración. A continuación se expone una imagen de la interfaz de creación de usuarios de Archiva:



The image shows the Apache Archiva web interface for creating an admin user. On the left is the Archiva logo and a search bar with 'Find', 'Search', and 'Browse' buttons. The main section is titled 'Create Admin User' and contains a form with the following fields: 'Username' (pre-filled with 'admin'), 'Full Name*' (empty), 'Email Address*' (empty), 'Password*' (empty), and 'Confirm Password*' (empty). A 'Create Admin' button is located at the bottom right of the form.

Figura 1.4 Interfaz para la creación de usuarios de Apache Archiva. Fuente: Elaboración propia.

Administración de Jenkins

Uno de los aspectos fundamentales de la administración de Jenkins es aplicar seguridad a los proyectos creados. Configurar la seguridad en Jenkins es fácil, es cuestión de ir a la página de configuración principal y seleccionar habilitar seguridad haciendo clic en el cuadro nombrado “Habilitar Seguridad”. A continuación se muestra una imagen donde se selecciona el modo de control de acceso, en este ejemplo se usa el modo “Usar Base de Datos de Jenkins”.

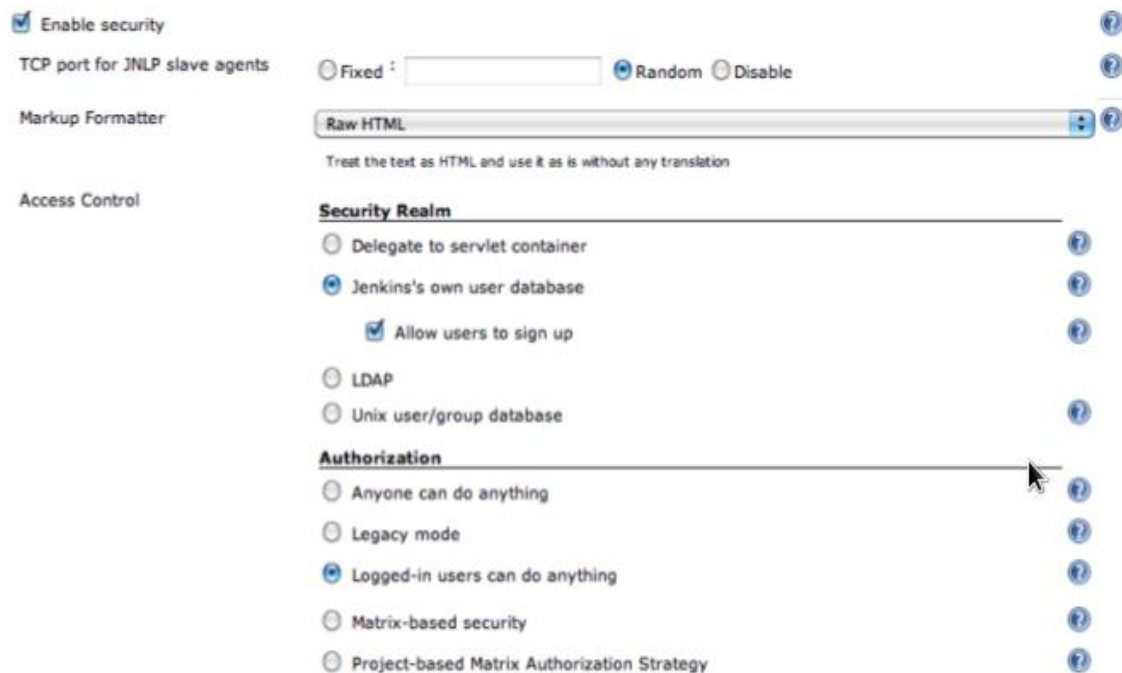


Figura 1.5 Interfaz para activación de la seguridad en Jenkins. Fuente: Elaboración propia.

Seguidamente en la sección de autorización se selecciona “Estrategia de seguridad para el proyecto” y se establece configuración que se mostrara seguidamente.


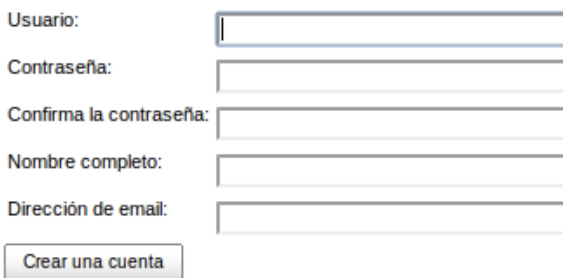
Usuario/Grupo	Global			Nodo				Tarea				Ejecutar		Vistas			Repositorio de software (SCM)			
	Administer	Read	RunScripts	Configure	Delete	Create	Disconnect	Connect	Create	Delete	Configure	Read	Build	Workspace	Delete	Update	Create	Delete	Configure	Tag
 admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Anónimo	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figura 1.6 Interfaz para configuración de la estrategia a seguir por el proyecto. Fuente: Elaboración propia.

Como se puede ver en la imagen anteriormente expuesta se define que existe un usuario o grupo llamado admin, el cual tiene todos los privilegios existentes, de esta manera se define cual va a ser el usuario de administración de Jenkins. También se puede notar que para los usuarios anónimos o sea, los que no se han logueado van a poder ver las tareas y los proyectos creados en Jenkins, estos van a poder descargar las distintas versiones generadas por las integraciones realizadas.

Una vez que existen los privilegios para el usuario de administración, solo falta crear dicho usuario. Para crear el usuario de administración de Jenkins es necesario hacer clic en el vínculo “Registrarse” que se encuentra en la parte superior derecha de la interfaz de Jenkins. Seguidamente aparece la interfaz por la cual se creará el usuario de administración, para el cual se definieron los permisos en el paso anterior, para esto es necesario llenar el siguiente formulario.

Crear una cuenta



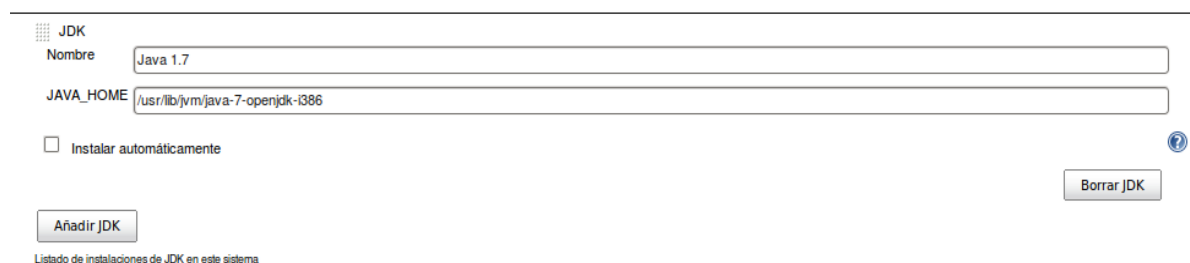
Formulario de creación de usuario en Jenkins. Incluye campos para:

- Usuario:
- Contraseña:
- Confirma la contraseña:
- Nombre completo:
- Dirección de email:

Botón: Crear una cuenta

Figura 1.7 Interfaz creación de usuarios en Jenkins. Fuente: Elaboración propia.

Otro aspecto importante es definir el JDK a usar por Jenkins, para esto es necesario abrir la página principal de configuración de Jenkins y en la sesión de JDK especificar el directorio principal del JDK conocido como JAVA_HOME. A continuación se mostrará un ejemplo de la interfaz donde se especifica esta configuración:



Interfaz de configuración del JDK en Jenkins. Muestra:

- Título: JDK
- Nombre: Java 1.7
- JAVA_HOME: /usr/lib/jvm/java-7-openjdk-1386
- Caja de verificación: ☐ Instalar automáticamente
- Botón: Añadir JDK
- Botón: Borrar JDK
- Texto: Listado de instalaciones de JDK en este sistema

Figura 1.8 Interfaz de configuración del JDK en Jenkins. Fuente: Elaboración propia.

También se necesita establecer las configuraciones necesarias para definir la herramienta de gestión y configuración de proyectos. En este caso se usará

Maven2. Para definir la versión de Maven a usar por Jenkins es necesario abrir la página principal de configuraciones de Jenkins en la sesión de Maven y especificar la dirección del directorio donde se encuentra instalado Maven, o sea el directorio MAVEN_HOME. A continuación se mostrará un ejemplo de la interfaz donde se especifica la configuración de la versión de Maven a usar:

Figura 1.8 Interfaz de configuración de la versión de Maven a usar por Jenkins. Fuente: Elaboración propia.

Una de las funcionalidades más importantes que tiene Jenkins es la facilidad que brinda para notificar a los usuarios tanto de problemas existente en los proyectos, resultado de las pruebas unitarias, fallas en el código y la estabilización de un código que anteriormente no funcionaba.

Para configurar las notificaciones de Jenkins es necesario abrir la página principal de configuración de Jenkins en la sesión de notificaciones y establecer la configuración que se mostrará a continuación.

Figura 1.9 Interfaz de configuración de las notificaciones de Jenkins. Fuente: Elaboración propia.

Como se puede apreciar en la imagen anterior es necesario establecer el servidor de correo saliente (SMTP), el sufijo de email por defecto y el correo a usar por Jenkins para crear las notificaciones.

Uno de los plug-in más usados por los desarrolladores en Jenkins es el que permite integrar Jenkins con Sonar, mediante el cual se pueden obtener datos estadísticos para mejorar la calidad del código. Para configurar sonar en Jenkins es necesario abrir la página principal de configuración en la sesión de Sonar y especificar los siguientes datos:

Sonar



Sonar installations	Name	<input type="text" value="SonarServer"/>
	Disable	<input type="checkbox"/>
		<small>Check to quickly disable Sonar on all jobs.</small>
	Server URL	<input type="text" value="http://10.208.7.202:8080/sonar/"/>
		<small>Default is http://localhost:9000</small>
	Server Public URL	<input type="text"/>
		<small>If not specified, then Server URL will be used</small>
	Database URL	<input type="text" value="jdbc:mysql://localhost:3306/sonar?autoReconnect=true&useUnicode=true&characterEncoding=utf8"/> 
		<small>Do not set if default embedded database.</small>
	Database login	<input type="text" value="root"/>
		<small>Default is sonar.</small>
	Database password	<input type="password" value="*****"/>
		<small>Default is sonar.</small>
	Database driver	<input type="text" value="com.mysql.jdbc.Driver"/> 
		<small>Do not set if you use the default embedded database on localhost.</small>
	Version of sonar-maven-plugin	<input type="text"/>
		<small>If not specified, then sonarsonar will be used.</small>
	Additional properties	<input type="text"/>
		<small>Additional properties to be passed to the mvn executable (example : -Dsome.property=some.value)</small>

Figura 1.10 Interfaz de configuración para la integración con Sonar. Fuente: Elaboración propia.

Administración de Sonar

Una vez concluida la instalación de Sonar, lo primero que se debe hacer es cambiar la contraseña que tiene por defecto sonar, para eso es necesario iniciar sesión en el sistema usando el usuario admin y contraseña admin, posteriormente

hacer clic en el vínculo administrator que parece en la parte superior derecha de la pantalla, el cual abre la interfaz de configuración del perfil del usuario actual.

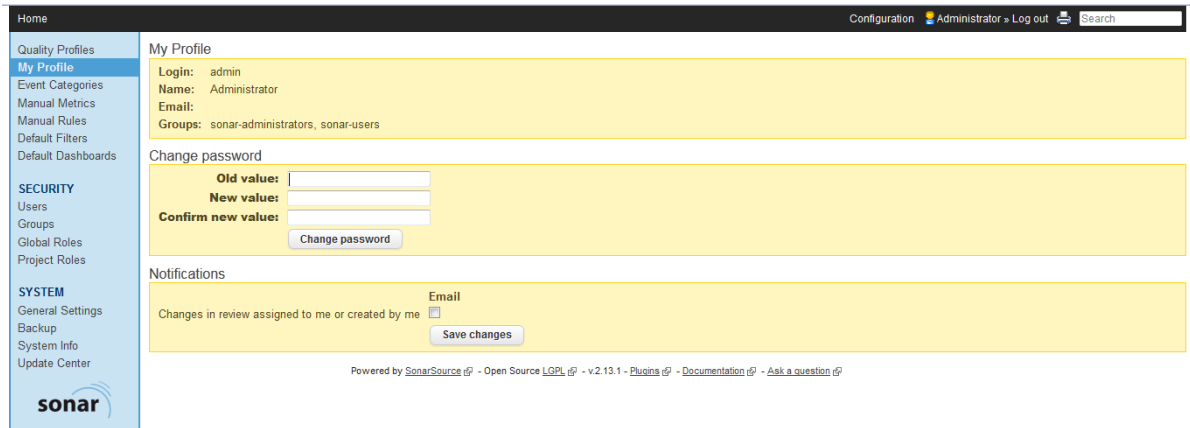


Figura 1.11 Interfaz de configuración del perfil de usuario en sonar. Fuente: Elaboración propia.

El próximo paso a realizar es establecer la URL base donde se encuentra alojado sonar. Para esto es necesario ir al menú de configuración, configuraciones generales, seguidamente aparece otro menú con las principales configuraciones de sonar, posteriormente se hace clic en el vínculo general y finalmente se establece la URL base donde se está alojado el servidor de Sonar. A continuación se muestra una imagen de la interfaz de configuración general de Sonar.

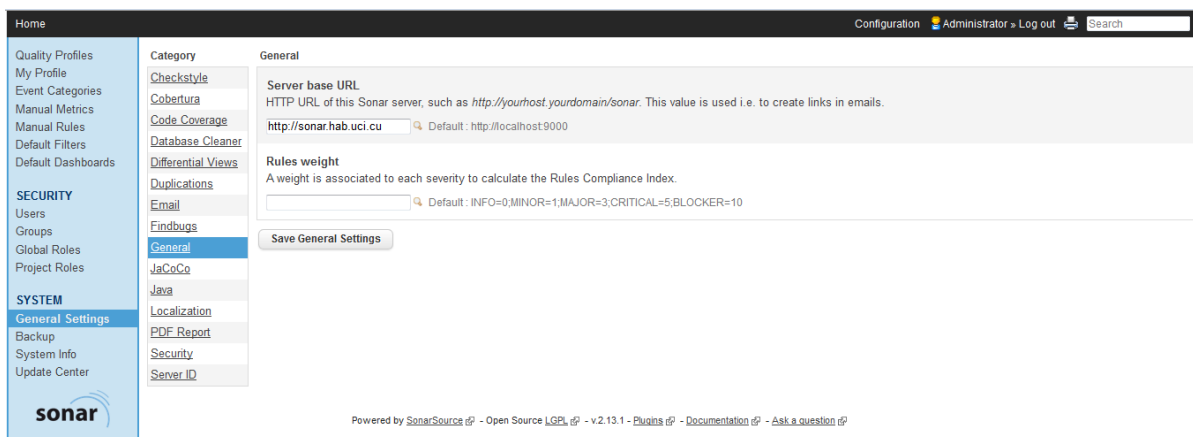


Figura 1.12 Interfaz de configuración general de sonar. Fuente: Elaboración propia.

Otro aspecto importante es configurar y activar el sistema de notificaciones por correo de Sonar. Para esto es necesario ir al menú de configuración,

configuraciones generales, seguidamente aparece otro menú con las principales configuraciones de sonar, posteriormente se hace clic en el vínculo Email y finalmente establecer las configuraciones necesarias para el correcto funcionamiento de las notificaciones. A continuación se expone un ejemplo de las configuraciones necesarias para las notificaciones con email usando el servidor de correo de la facultad.

The screenshot shows the Sonar Configuration interface. On the left is a sidebar with a 'sonar' logo and a menu including 'Quality Profiles', 'My Profile', 'Event Categories', 'Manual Metrics', 'Manual Rules', 'Default Filters', 'Default Dashboards', 'SECURITY' (with sub-items 'Users', 'Groups', 'Global Roles', 'Project Roles'), 'SYSTEM' (with sub-items 'General Settings', 'Backup', 'System Info', 'Update Center'), and 'General Settings' (which is highlighted). The main content area is titled 'Email Settings' and contains the following fields and options:

- SMTP host:** 10.208.0.44 (with a note: "For example 'smtp.gmail.com'. Leave blank to disable email sending.")
- SMTP port:** 25 (with a note: "Port number to connect with SMTP server.")
- Use secure connection:** No (with a note: "Whether to use secure connection and its type.")
- SMTP username:** (with a note: "Optional - if you use authenticated SMTP, enter your username.")
- SMTP password:** (with a note: "Optional - as above, enter your password if you use authenticated SMTP.")
- From address:** sonar@hab.uci.cu (with a note: "Emails will come from this address. For example - 'noreply@sonarsource.com'. Note that server may ignore this setting (like does Gmail).")
- Email prefix:** SONAR (with a note: "This prefix will be prepended to all outgoing email subjects.")

Below these fields is a 'Save Email Settings' button. Under the 'Test Configuration' section, there are fields for 'To:', 'Subject: Test Message from Sonar', and a 'Message: This is a test message from Sonar' text area. A 'Send Test Email' button is located at the bottom of this section.

Figura 1.13 Interfaz de configuración para las notificaciones por email de Sonar. Fuente: Elaboración propia.

El siguiente paso es la instalación del plug-in para la generación de reportes en formato PDF. Para esto primeramente es necesario configurar el método de acceso a internet. En este caso se establecerán las configuraciones necesarias para que Sonar acceda a internet mediante un proxy. Para esto es necesario editar el fichero que se encuentra en el directorio /home/usuario/SONAR_HOME/conf/sonar.properties, y en la sección UPDATE CENTER y establecer las configuraciones necesarias para la conexión a internet mediante un proxy. A continuación se muestra un ejemplo de la configuración mencionada anteriormente.

```

#-----
# UPDATE CENTER
#-----

# The Update Center requires an internet connection to request http://update.sonarsource.org

# It is activated by default:

#sonar.updatecenter.activate=true

# HTTP proxy (default none)

http.proxyHost=10.208.0.2

http.proxyPort=3128

# NT domain name if NTLM proxy is used

#http.auth.ntlm.domain=

# SOCKS proxy (default none)

#socksProxyHost=

#socksProxyPort=

# proxy authentication. The 2 following properties are used for HTTP and SOCKS proxies.

http.proxyUser=usuario

http.proxyPassword=password

```

Por último se procede a instalar el plug-in para la generación de reportes en formato PDF. Para esto es necesario ir al menú de configuración, al centro de actualización, seguidamente se hace clic en la pestaña de plug-ins disponibles, se buscan los plug-ins de visualización y reportes y se procede a instalar el plug-in PDF Report. A continuación se muestra una imagen de la interfaz de instalación del plug-in para generar reportes en formato PDF.



Figura 1.14 Interfaz para la instalación de plug-ins en Sonar. Fuente: Elaboración propia.

Conclusiones del capítulo

En este capítulo se analizó el estado actual del proceso de desarrollo de proyectos con el entorno de `WebSocket` y los principales problemas existentes en este. Se analizó la encuesta realizada con el objetivo de diagnosticar los problemas existentes en los proyectos desarrollados en la facultad con el marco de trabajo de `WebSocket`. Una vez diagnosticado el problema se procedió a crear y detallar el principio de funcionamiento de la propuesta de solución. Seguidamente se describe el proceso de instalación y configuración de las herramientas seleccionadas para la solución del problema. Finalmente se describen las configuraciones principales y la administración de las herramientas que componen la solución.

CAPITULO 3. VALIDACIÓN DE LA PROPUESTA.

En el desarrollo de este capítulo se procede a validar la propuesta del entorno de integración continua para los proyectos creados con el marco de trabajo de jWebSocket haciendo uso de método de validación por criterio de especialistas. En este capítulo se explica detalladamente el proceso de selección de los especialistas, la conformación de un cuestionario para la validación de la solución. También se realizará un análisis porcentual de las respuestas a cada una de las preguntas que conforman el cuestionario realizado para la validación de la solución. Posteriormente se expondrá un resumen general de todas las respuestas para cada una de los especialistas entrevistados. Finalmente se dará a conocer el aporte social y económico brinda la propuesta de entorno de integración continua realizada.

3.1. Método de validación de la propuesta

Para la validación de la propuesta de del entorno de integración continua para proyectos desarrollados con el marco de trabajo jWebSocket se utilizó el criterio de un grupo de especialistas. Este panel se conformó con especialistas que dominan las principales características de un entorno de integración continua. La eficiencia de los resultados depende en su totalidad de la calidad del diseño de las preguntas de los cuestionarios realizados y de la selección de los especialistas para una correcta validación de la solución.

El procedimiento a seguir para la validación de la solución consta de tres fases principales.

- **Fase preliminar:** Se delimita el contexto, los objetivos, el diseño, los elementos básicos del trabajo y la selección del panel de los especialistas.
- **Fase exploratoria:** En esta fase se procede a realizar la creación y aplicación de los cuestionarios a los especialistas seleccionados en la fase anterior.

- **Fase final:** en esta fase se procede a analizar y presentar la información de los resultados de la encuesta realizada.

3.2. Selección de los especialistas para la validación

Se entiende por especialista al individuo capaz de ofrecer una validación conclusiva de un problema dado y hacer recomendaciones respecto a sus momentos fundamentales con un máximo de competencia. Las características de los especialistas seleccionados influyen decisivamente en la calidad de los resultados obtenidos. Estas características son las capacidades de calificación técnica, capacidad de emitir una decisión al respecto, previos conocimientos sobre el tema a evaluar, disposición a participar en la evaluación de la solución, entre otros.

La calidad de los resultados obtenidos dependerá totalmente de una buena selección de los especialistas atendiendo a las características de estos, junto a otras cualidades que estos deben tener como son la sinceridad, la responsabilidad, entre otras, las cuales hacen que sus calificaciones sean más confiables y válidas para el objetivo propuesto. Para el desarrollo de este proceso se tiene tres etapas principales:

Establecer la cantidad de especialistas: Para la validación de este trabajo se decide seleccionar seis especialistas teniendo en cuenta el nivel de profundidad y complejidad del contenido. Dichos especialistas son los líderes de los principales proyectos de la facultad, los cuales lideran los proyectos desde sus inicios por lo tanto tiene un amplio conocimiento sobre aspectos que rigen la calidad del producto.

Una buena elección de los especialistas tiene consigo obtener mejores resultados con calidad y mayor confiabilidad en las opiniones brindadas.

Conformar el listado de los especialistas: Para la realización del listado de especialistas se tuvo en cuenta posibilidad real de participación en proyectos productivos. Se determinó que los especialistas a consultar deben ser

profesionales de la UCI, con conocimientos acerca de la integración continua para que puedan emitir un criterio efectivo sobre la propuesta.

Para la selección de los especialistas se tuvo en cuenta varios aspectos como son:

- Disposición a participar en la encuesta
- Ser graduados en la especialidad en ingeniería en ciencias informáticas.
- Poseer conocimientos básicos de sistemas de integración continua.
- Estar vinculados a proyectos reales.

Confirmar la participación de los especialistas: Una vez creado el listado, se invitó personalmente a cada especialista elegido para participar en la evaluación. Allí se les explicó en qué consistía el trabajo en general, la propuesta a evaluar y el objetivo de la realización de la encuesta, así como el plazo de entrega. Una vez recibida la respuesta positiva, se estableció el listado final de los especialistas, informando a cada especialista su inclusión en el proceso a evaluar y las instrucciones necesarias para contestar las preguntas. De esta forma, culmina el proceso de selección, logrando la participación de los seis especialistas seleccionados.

3.3. Elaboración de la encuesta para la validación de la solución

En la encuesta se realizaron ocho preguntas principales, las cuales tienen como objetivo medir la calidad y el impacto causado por la propuesta de solución. La primera pregunta tiene como objetivo medir la repercusión que tendrá la propuesta de solución sobre el desarrollo de los proyectos de la facultad. Posteriormente se realizan preguntas con el objetivo de conocer si la solución resolverá los problemas existentes actualmente con los proyectos desarrollados con el marco de trabajo jWebSocket, problemas como integración de códigos, generación de versiones, si se logra economizar el tiempo de trabajo de los desarrolladores. También se pregunta sobre si aumentará la calidad del código

gracias a la integración con Sonar. Se realizan preguntas como la posibilidad de aceptación por parte de los usuario, o sea si la solución tiene posibilidad de ser usada por los proyectos. Por último se pregunta si la propuesta de solución tiene posibilidades de aumentar la productividad en los proyectos y evitar el retraso de la entrega final de un producto.

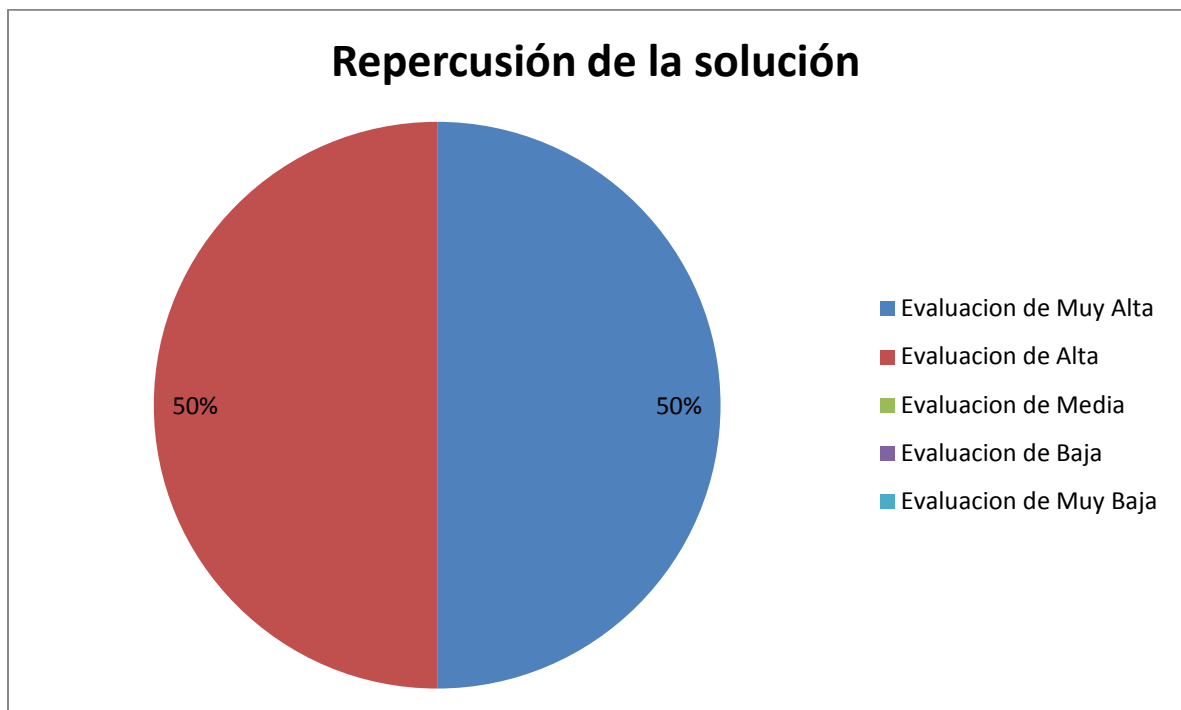
3.4. Resultados de la validación

Para el análisis y procesamiento de los resultados de la propuesta de entorno de integración continua para el marco de trabajo de jWebSocket se cogió un rango de evaluación de 1 a 5 para cada una de las preguntas de la encuesta realizada, donde 1 es el mínimo y 5 el máximo. A continuación se mostrará una tabla con los resultados de la encuesta para cada uno de los especialistas.

	Especialista 1	Especialista 2	Especialista 3	Especialista 4	Especialista 5	Especialista 6
Pregunta 1	4	4	5	4	5	5
Pregunta 2	5	5	5	4	4	5
Pregunta 3	4	4	4	4	5	4
Pregunta 4	5	5	4	4	5	5
Pregunta 5	5	5	5	5	5	5
Pregunta 6	4	5	4	5	5	5
Pregunta 7	5	5	4	5	5	5
Pregunta 8	4	4	5	5	5	5

Tabla 2 Resultados de la encuesta para la validación de la solución.

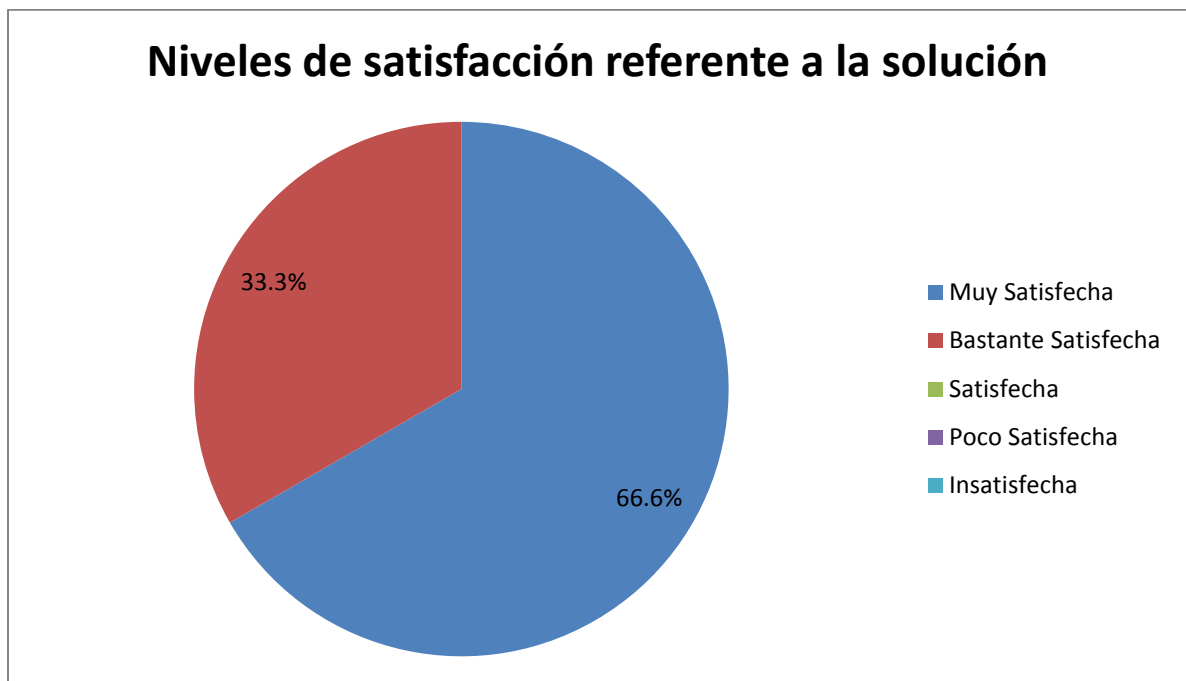
A continuación se procederá a analizar cada una de las preguntas de la encuesta realizada. Se comenzará por conocer el criterio de los especialistas seleccionados en cuanto a la repercusión que tendrá la solución para el desarrollo de proyectos. A continuación se mostrará una gráfica con los resultados de esta pregunta:



Gráfica 1 Repercusión de la solución planteada. Fuente: Elaboración propia.

Como se puede observar en la gráfica anteriormente expuesta el 50% de los especialistas encuestados dan una evaluación de muy alta y el otro 50% de los encuestados una evaluación de alta, lo que tiene como resultado un buen criterio por parte de los especialistas en cuanto a la repercusión que tendrá la solución para su uso en proyectos reales.

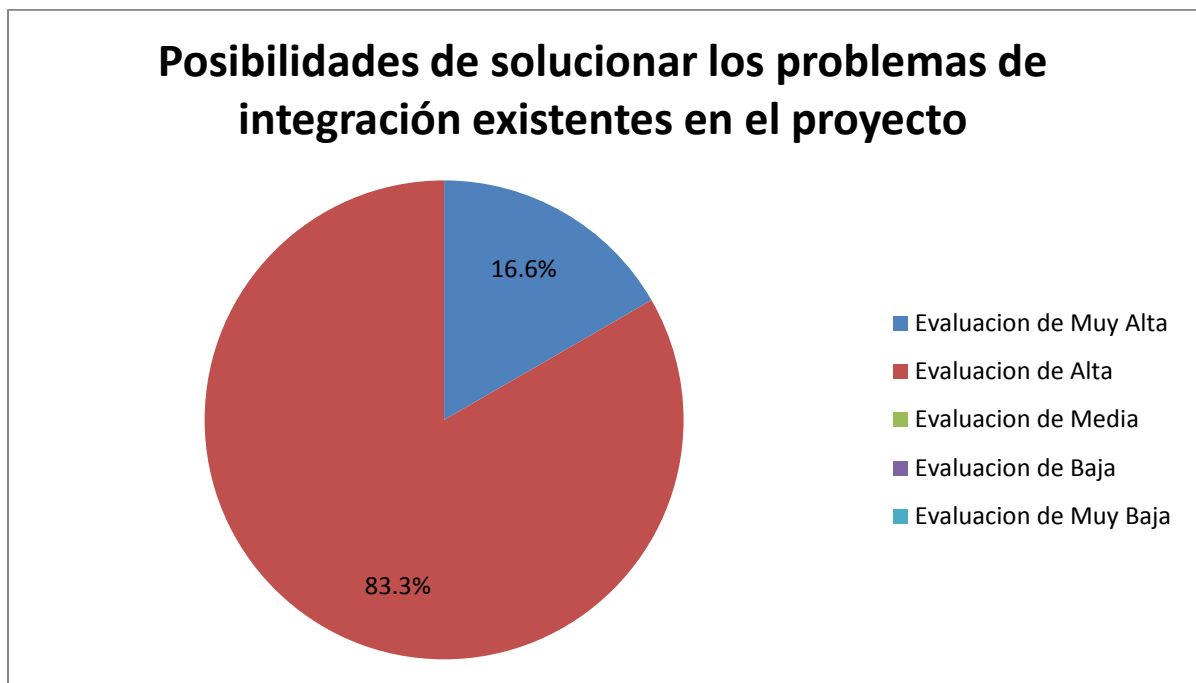
A continuación se mostrará una gráfica referente a las respuestas de los especialistas en cuanto a la satisfacción de las necesidades por parte de la solución:



Gráfica 2 Niveles de satisfacción de las necesidades por parte de la solución. Fuente: Elaboración propia.

Como se puede observar en la gráfica anteriormente expuesta un 66.6% de los especialistas encuestados dan una evaluación de muy alta y el otro 33.3% de los encuestados una evaluación de alta, lo que tiene como resultado un buen criterio por parte de los especialistas referente a la satisfacción de las necesidades por parte de la solución.

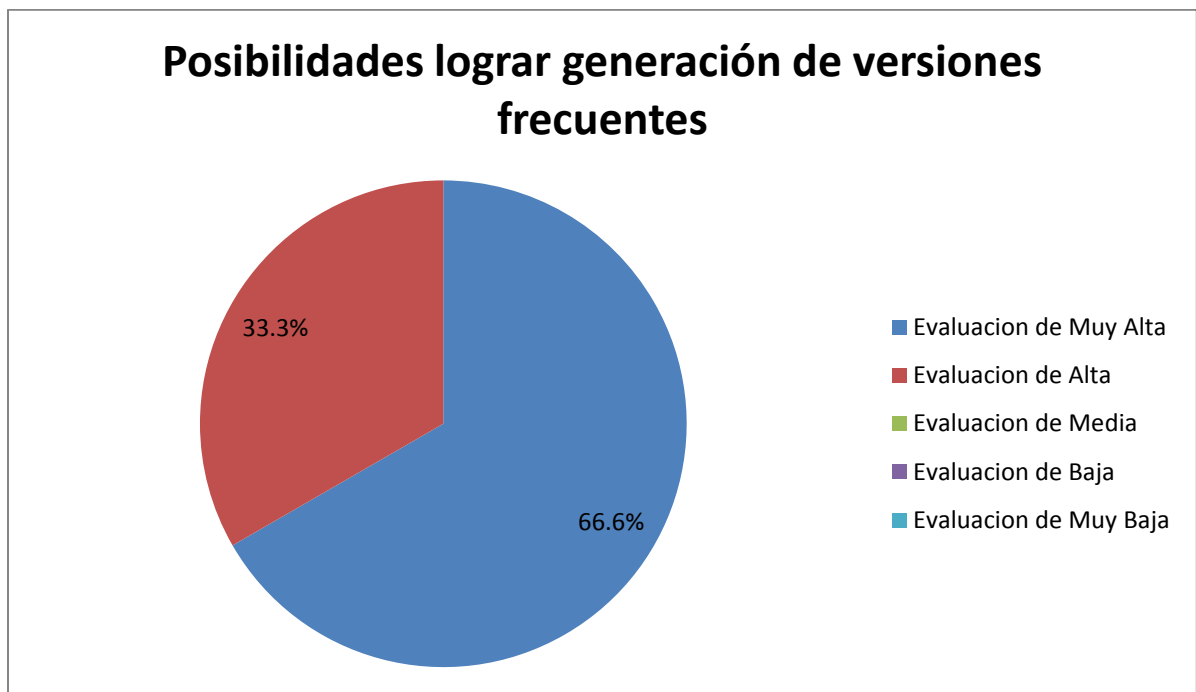
A continuación se mostrará una gráfica referente a las respuestas de los especialistas en cuanto a las posibilidades que tiene la propuesta de solución para resolverlos problemas de integración de códigos que existe actualmente en los proyectos desarrollados con el marco de trabajo de `jQuery`:



Gráfica 3 Resultado de las posibilidades de solucionar problemas de integración de códigos. Fuente: Elaboración propia.

Como se puede observar en la gráfica anteriormente expuesta un 83.3% de los especialistas encuestados dan una evaluación de muy alta y el otro 16.6% de los encuestados una evaluación de alta, lo que tiene como resultado un buen criterio por parte de los especialistas referente a las posibilidades por parte de la solución de resolver los problemas de integración de códigos existente actualmente en los proyectos desarrollados con el marco de trabajo de `jQueryWebSocket`.

A continuación se mostrará una gráfica referente a las respuestas de los especialistas en cuanto a las posibilidades lograr una seguida generación de versiones en los proyectos desarrollados con el marco de trabajo de `jQueryWebSocket`:

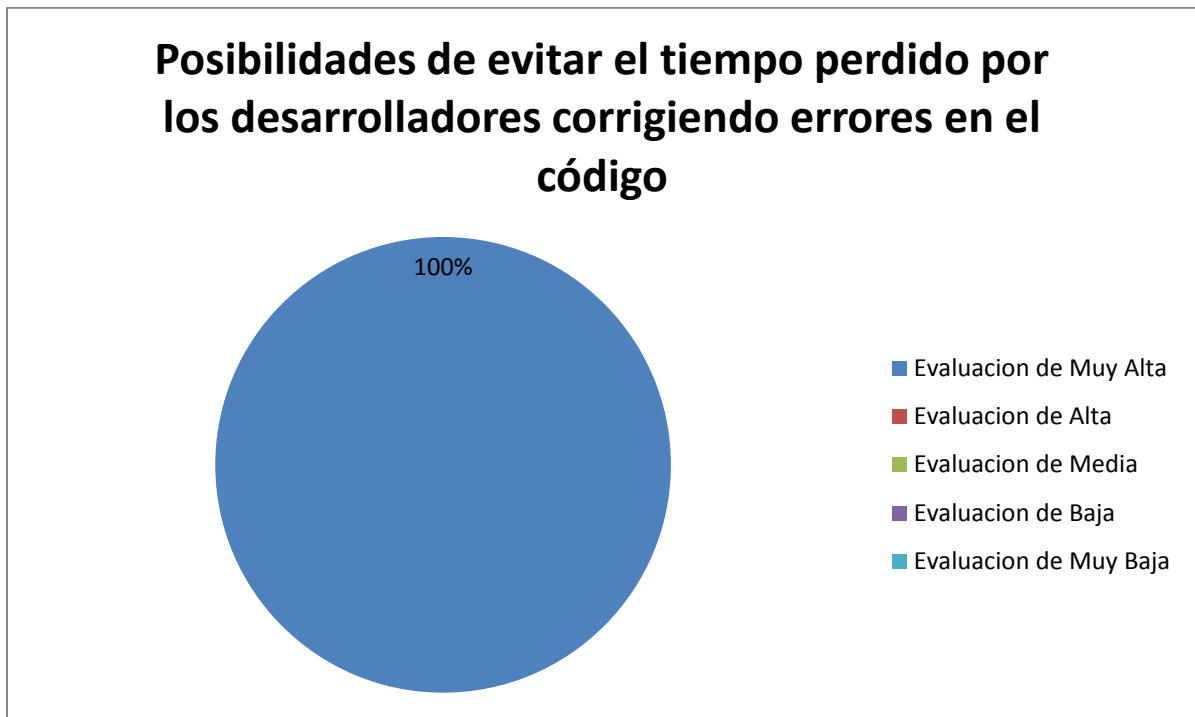


Gráfica 4 Resultado de las posibilidades de lograr una frecuente generación de versiones.

Fuente: Elaboración propia.

Como se puede observar en la gráfica anteriormente expuesta un 66.6% de los especialistas encuestados dan una evaluación de muy alta y el otro 33.3% de los encuestados una evaluación de alta, lo que tiene como resultado un buen criterio por parte de los especialistas referente a las posibilidades lograr una frecuente generación de versiones en los proyectos desarrollados con el marco de trabajo jWebSocket por parte de la solución.

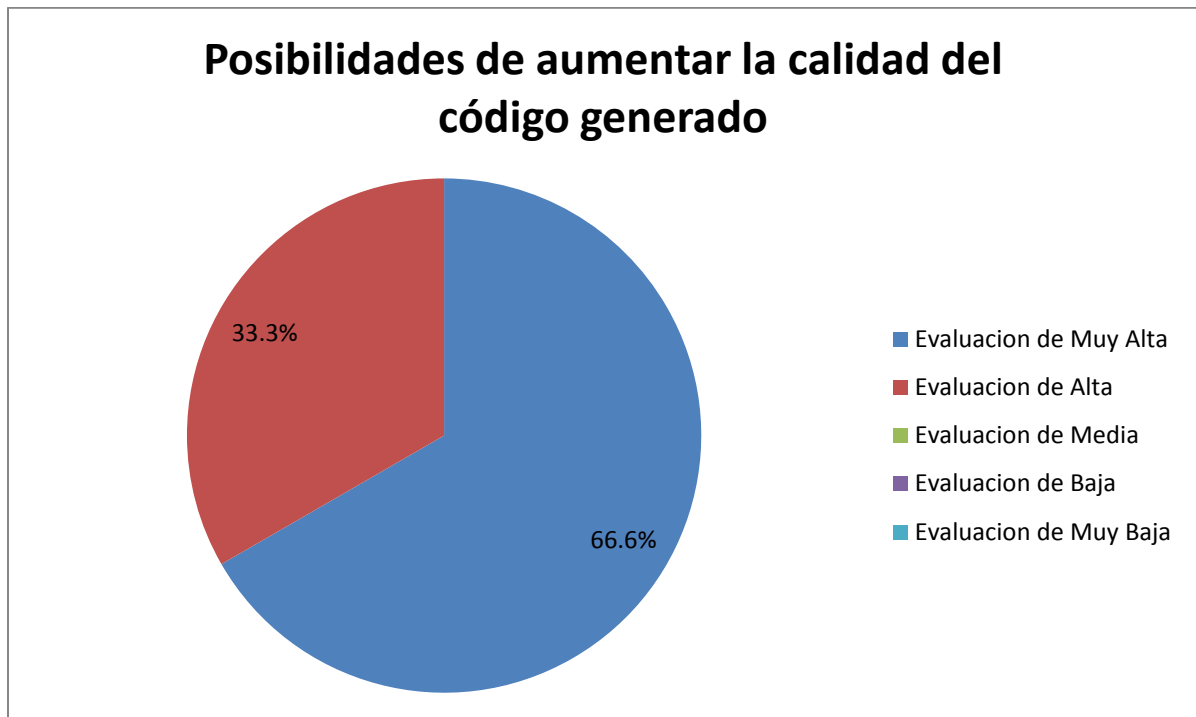
A continuación se mostrará una gráfica referente a las respuestas de los especialistas en cuanto a las posibilidades lograr evitar el tiempo perdido por parte de los desarrolladores corrigiendo errores en el código:



Gráfica 5 Resultado de las posibilidades de evitar el tiempo perdido corrigiendo errores en el código por parte de los desarrolladores. Fuente: Elaboración propia.

Como se puede observar en la gráfica anteriormente expuesta, se logra un 100% por parte de los especialistas encuestados con la evaluación de muy alta para la pregunta que responde a las posibilidades de evitar el tiempo perdido por parte de los desarrolladores corrigiendo errores en el código. Este es uno de los aspectos más importantes que se quiere lograr con la solución, debido que la esto atenta contra la entrega en tiempo de un producto dado.

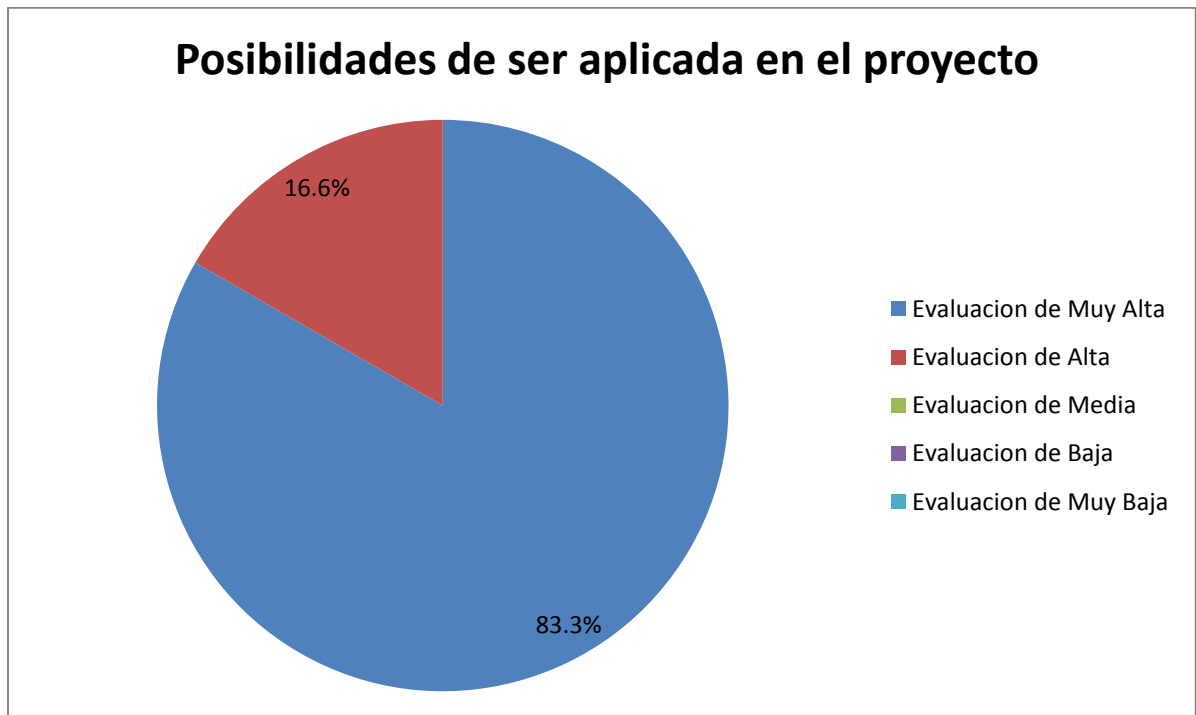
A continuación se mostrará una gráfica referente a las respuestas de los especialistas en cuanto a las posibilidades de aumentar la calidad del código generado por parte de los desarrolladores, esto se debe a la integración del servidor de integración continua Jenkins con el servicios de métricas Sonar:



Gráfica 6 Resultado de las posibilidades de aumentar la calidad del código generado por parte de los desarrolladores. Fuente: Elaboración propia.

Como se puede observar en la gráfica anteriormente expuesta un 66.6% de los especialistas encuestados dan una evaluación de muy alta y el otro 33.3% de los encuestados una evaluación de alta, lo que tiene como resultado un buen criterio por parte de los especialistas referente a las posibilidades mejorar la calidad de los códigos generados por parte de los desarrolladores para proyectos desarrollados con el marco de trabajo jWebSocket.

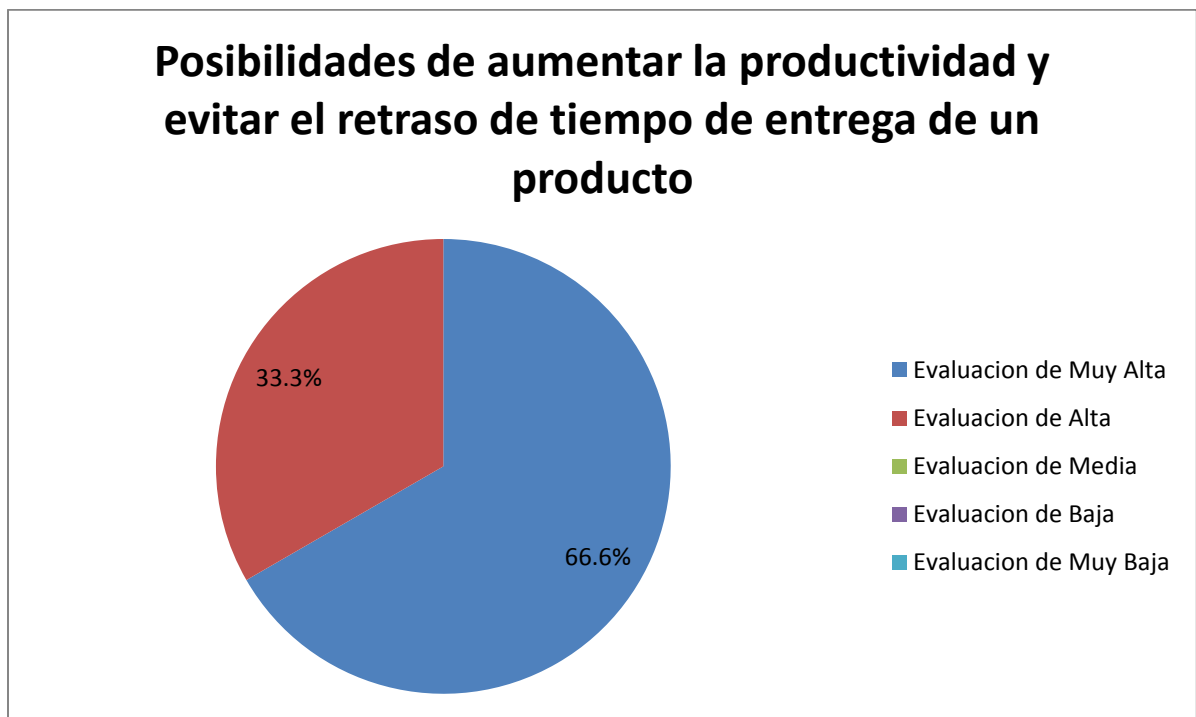
A continuación se mostrará una gráfica referente a las respuestas de los especialistas en cuanto a las posibilidades aplicar la solución propuesta en proyectos reales:



Gráfica 7 Resultado de las posibilidades de que la solución propuesta sea aplicada en proyectos reales. Fuente: Elaboración propia.

Como se puede observar en la gráfica anteriormente expuesta un 83.3% de los especialistas encuestados dan una evaluación de muy alta y el otro 16.6% de los encuestados una evaluación de alta, lo que tiene como resultado un buen criterio por parte de los especialistas referente a las posibilidades de aplicar la solución propuesta en proyectos reales.

A continuación se mostrará una gráfica referente a las respuestas de los especialistas en cuanto a las posibilidades que tiene la solución de aumentar la productividad y evitar el retraso en el tiempo de entrega de un producto dado:

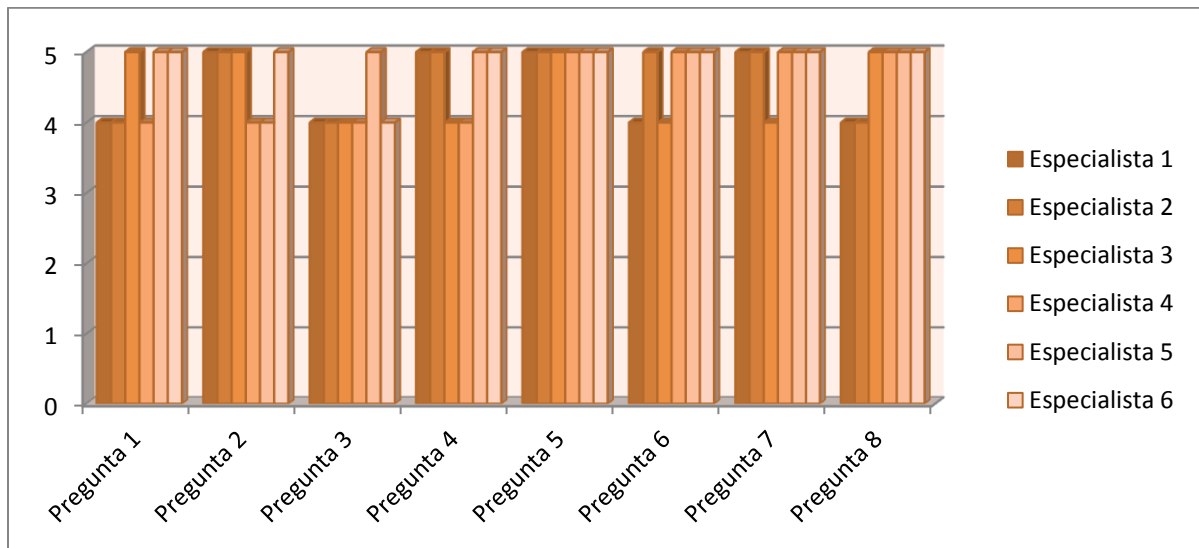


Gráfica 8 Resultado de las posibilidades de aumentar la productividad y evitar el retraso de los tiempos de entrega de los productos. Fuente: Elaboración propia.

Como se puede observar en la gráfica anteriormente expuesta un 66.6% de los especialistas encuestados dan una evaluación de muy alta y el otro 33.3% de los encuestados una evaluación de alta, lo que tiene como resultado un buen criterio por parte de los especialistas referente a las posibilidades aumentar la productividad y evitar el retraso de los tiempos de entrega de los proyectos realizados con el marco de trabajo jWebSocket.

De forma general los especialistas dieron un alto valor a cada una de las respuestas de las preguntas para la validación de la propuesta de solución desarrollada ya que en su evaluación los resultados oscilaron entre 4 y 5 puntos respectivamente lo que representa un promedio de 4.64 puntos.

Por último se mostrará una gráfica donde se pueden observar las respuestas de cada uno de los especialistas entrevistados para cada pregunta realizada.



Gráfica 9 Respuestas de pregunta por especialistas. Fuente: Elaboración propia.

3.5. Aporte Social y Económico

Con la evolución de la informática y las TIC se ha desarrollado a su vez la Ingeniería de Software y la Gestión de Proyectos Informáticos. Estas disciplinas tienen entre sus principales metas establecer buenas prácticas en el desarrollo de proyectos de software para lograr mayor calidad de los productos, tiempos cortos y menor esfuerzo. La presente investigación tiene como objetivo proponer un entorno de integración continua para el desarrollo de proyectos de software con el marco de trabajo jWebSocket. Esta propuesta permite mejorar el proceso de integración y calidad del código fuente en un entorno de desarrollo de software. Por ello garantiza disminuir el tiempo de finalización de los productos informáticos.

El entorno de integración para el marco de trabajo jWebSocket garantiza mayor precisión y organización del proceso de gestión de cambios y generación de versiones. Estos factores influyen también favorablemente en lograr menor esfuerzo del equipo de desarrollo y menor tiempo ante la ocurrencia de cambios importantes en el proyecto.

El desarrollo de software y el uso de las TIC son el sustento de la actual sociedad de la información. Cada día se hace necesario que el desarrollo de software logre

niveles más eficientes. El resultado de la presente investigación aporta un entorno de integración continua que logra un impacto importante al desarrollo de software. Con su utilización en el desarrollo de aplicaciones jWebSocket garantiza que la comunidad de desarrolladores a nivel internacional mejoren sus niveles de eficiencia y calidad de sus productos.

El aporte económico de la investigación se encuentra muy vinculado al impacto social de la propuesta. Al garantizar menores tiempos, menor esfuerzo del equipo de desarrollo y mayor calidad de los productos esto permite obtener mayores ganancias económicas. Aumentar las utilidades en el desarrollo de software con el marco de trabajo jWebSocket beneficia de esta manera a empresas que usen esta tecnología en el desarrollo de las aplicaciones informáticas y a desarrolladores individuales que forman parte de la comunidad jWebSocket.

Conclusiones del capítulo

En este capítulo se procedió a validar la propuesta de solución mediante el uso del método de validación por criterio de especialistas. Primeramente se explicó en que consiste el método seleccionado para la validación de la solución, se expuso las tres fases principales para realizar la validación. Posteriormente se explica cómo se realiza la selección del panel de especialistas y las características que se tuvieron en cuenta para la selección de estos. También se dio una breve explicación de la elaboración de la encuesta para realizar la validación de la solución. Finalmente se analizó los resultados de las respuestas del cuestionario realizado y se hizo un análisis detallado por cada una de las respuestas realizadas en la encuesta. Se puede observar que la solución planteada tiene una buena evaluación por parte de los especialistas seleccionado en cada uno de los aspectos evaluados.

CONCLUSIONES

- El análisis de la bibliografía consultada, permitió fundamentar teóricamente a la integración continua como práctica de desarrollo de software que aparece con la creación de las metodologías ágiles, donde los miembros de un equipo integran su código como mínimo una vez al día, generando por cada integración realizada una nueva versión del proyecto.
- No existe actualmente una integración automática de los códigos de los distintos desarrolladores en proyectos creados con el marco de trabajo `jQueryWebSocket`.
- Se propuso un entorno de integración continua formado por las herramientas: JUnit, Subversion, Maven, Apache Archiva, Sonar y Jenkins.
- Se validó la propuesta de solución haciendo uso el método de validación por criterio de especialistas, obteniendo una evaluación positiva de la solución propuesta.

RECOMENDACIONES

- Continuar estudiando los temas relacionados con la Integración Continua.
- Aplicar la propuesta solución en proyectos con entornos de trabajos similares a jWebSocket.
- Extender las funcionalidades de la solución para dar soporte a otros lenguajes.
- Profundizar en la integración de Jenkins con Sonar y sus complementos, con el objetivo de aumentar las funcionalidades de estos dos servidores.

BIBLIOGRAFÍA

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

Apache. 2012. Apache Subversion. [En línea] 2012. <http://subversion.apache.org/features.html>.

Archiva, Apache. 2012. Apache Archiva. [En línea] 2012. <http://archiva.apache.org/>.

Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato. 2004. *Control de versiones con Subversion*. 2004.

Bisset, Clenda Perez. 2009. *Herramienta para la administracion de repositorios Subversion (PhpSvnAdmin)*. 2009. Tesis.

Carballo, Leamny Teresa Fernandez. 2007. *Estrategia metodologica para el desarrollo de Software de Gestion a Distancia basado en Programacion Extrema*. 2007. Tesis.

Continuum, Apache. 2012. Apache Continuum. [En línea] 2012. <http://continuum.apache.org/>.

Figuerola, 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>.

Fowler, Martin. 2006. Integración Continua. [En línea] Mayo de 2006. <http://martinfowler.com/articles/continuousIntegration.html>.

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.

Grant, Yassef Amed Galarraga. 2010. *Propuesta de entorno de integración continua para el desarrollo de software en el Centro de Informatización Universitaria.* 2010. Tesis.

Guilarte, Alianis La Hoz. 2007. *Pruebas para el software Administracion Contable de los Registros y Notarias de la Republica Bolivariana de Venezuela.* 2007. Tesis.

Hudson. 2012. Hudson. [En línea] 2012. <http://hudson-ci.org/>.

Javier Tuya, Isabel Ramos Román, Javier Dolado Cosín. 2007. *Técnicas Cuantitativas para la gestión en la ingeniería del software.* 2007.

jUnit. 2012. jUnit. [En línea] 2012. www.junit.org.

Ludisley la Torre Hernández, Mariela Cepero Nuñez. 2007. *Propuesta de Metricas para Perfeccionar la Gestion de la Calidad en los Procesos de Desarrollo de Software.* 2007. Tesis.

Miranda, Roberto Arjona. 2008. *Estudio y propuesta de metodologias de desarrollo para los proyectos de la Facultad 1 segun su alcance.* 2008. Tesis.

Reyes, Dainys Gainza. 2007. *Documentacion para los flujos de trabajo de diseño e implementacion de software de gestion para la UCI.* 2007. Tesis.

Rodriguez, Leticia Garcia. 2010. *Repositorio de Contenidos para el Diccionario Enciclopedico Libertad.* 2010. Tesis.

Schulze, Alexander. 2011. jWebSocket. [En línea] 2011. <http://enapso.org/?q=node/205>.

Smart, John Ferguson. 2011. *Continuous Integration for the Masses.* 2011.

Sonar. 2012. Sonar. [En línea] 2012. <http://www.sonarsource.org/>.

Sonatype. 2012. Sonatype. [En línea] 2012. <http://www.sonatype.com/>.

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.

TestGN. 2012. TestGN. [En línea] 2012. <http://testng.org/doc/index.html>.

ANEXOS

Anexo 1

Encuesta para la determinación del diagnóstico del problema.

1. ¿Cómo valora los problemas de integración de cada uno de los códigos de los diferentes desarrolladores?

___Muy Alta

___Alta

___Media

___Baja

___Muy baja

2. ¿Considera usted que exista una seguida generación de versiones del proyecto desarrollado?

___Muy Alta

___Alta

___Media

___Baja

___Muy baja

3. ¿Considera usted que existe una seguida realización de pruebas al código?

___Muy Alta

___Alta

___Media

___Baja

___Muy baja

4. ¿Considera usted que existe una pérdida de tiempo por parte de los desarrolladores arreglando defectos en el código?

___Muy Alta

___Alta

___Media

___Baja

___Muy baja

5. ¿Considera usted que existan problemas de entrega en tiempo y finalización de los proyectos?

___Muy Alta

___Alta

___Media

___Baja

___Muy baja

Anexo 2

Encuesta para la validación de la propuesta de entorno de integración continua.

1. ¿Cree usted que la solución planteada tenga una repercusión positiva en el desarrollo del proyecto?

___Muy Alta

___Alta

___Media

___Baja

___Muy baja

2. ¿Considera usted que la propuesta de solución satisfaga las necesidades del proyecto?

___Muy satisfecha

___Bastante satisfecha

___Satisfecha

___Poco satisfecha

___Insatisfecha

3. ¿Considera usted que la propuesta de solución tiene posibilidades de solucionar los problemas de integración existentes en el proyecto?

___Muy Alta

___Alta

___Media

___Baja

___Muy baja

4. ¿Considera usted que la propuesta de solución tiene posibilidades lograr generación de versiones frecuentes en el proyecto?

___Muy Alta

___Alta

___Media

___Baja

___Muy baja

5. ¿Considera usted que la propuesta de solución tiene posibilidades de evitar el tiempo perdido por los desarrolladores corrigiendo errores en el código?

___Muy Alta

___Alta

___Media

___Baja

___Muy baja

6. ¿Considera usted que la propuesta de solución tiene posibilidades de aumentar la calidad del código generado por los diferentes desarrolladores?

___Muy Alta

___Alta

___Media

___Baja

___Muy baja

7. ¿Considera usted que la propuesta de solución tiene posibilidades de ser aplicada en el proyecto?

___Muy Alta

___Alta

___Media

___Baja

___Muy baja

8. ¿Considera usted que la propuesta de solución tiene posibilidades de aumentar la productividad y evitar el retraso de tiempo de entrega de un producto?

___Muy Alta

___Alta

___Media

___Baja

___Muy baja

GLOSARIO DE TÉRMINOS

XP: Metodología de desarrollo de software.

Token: Pequeños grupos de datos que representan un conjunto de información.

JSON: (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.

XML: (eXtensible Markup Language - lenguaje de marcas extensible) es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

CSV: (comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla.

Plug-in: Un plug-in o complemento es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

AJAX: (Asynchronous JavaScript And XML) es una técnica de desarrollo web para crear aplicaciones interactivas.

TDD: Desarrollo guiado por pruebas, o Test-driven development (TDD) es una práctica de programación que involucra otras dos prácticas: Escribir las pruebas primero (Test First Development) y Refactorización.

GNU: (licencia GPL) más conocida por sus siglas en inglés: GNU General Public License.

GPL: La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente sus siglas del inglés GNU GPL, es una licencia creada por la Free Software Foundation en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución.

SQL: El lenguaje de consulta estructurado o SQL (por sus siglas en inglés structured query language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas.

LDAP: (Lightweight Directory Access Protocol, Protocolo Ligero de Acceso a Directorios) es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

Commit: Es cuando subes los cambios del trabajo al repositorio de control de versiones.

.NET: es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

PHP: es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.

HTML: siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web.

CSS: Las hojas de estilo en cascada (en inglés Cascading Style Sheets), CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML.

LGPL: La Licencia Pública General Reducida de GNU, o más conocida por su nombre en inglés GNU Lesser General Public License (antes GNU Library General Public License o Licencia Pública General para Bibliotecas de GNU), o simplemente por su acrónimo del inglés GNU LGPL es una licencia de software.

PL/SQL: Lenguaje de programación incrustado en Oracle y PostgreSQL.