INTELLIJ IDEA 2018.3.1 y GITHUB como herramientas para el control de cambios en proyectos de equipos de desarrollo distribuidos

Loja Mora Nancy Magaly, Molina Ríos Jimmy Rolando, Loja Mora Fausto, Cañarte Vega Erick Antonio
nmloja@utmachala.edu.ec, jmolina@utmachala.edu.ec, faustol@gmail.com, ecanarte_est@utmachala.edu.ec

RESUMEN:

El desarrollo de proyectos informáticos en equipos de desarrollo distribuidos no fuera posible hoy en día, gracias a herramientas que permiten el desenvolvimiento adecuado para que estos sean finalizados a tiempo. En la actualidad existen muchas herramientas de desarrollo que facilita esta labor, sin embargo debido a el surgimiento de nuevas tecnologías, muchas aún no han sido exploradas. El presente trabajo de investigación se centra en la configuración de un entorno de trabajo para el control de cambios en proyectos de equipos de desarrollo distribuidos. La metodología del presente trabajo es experimental, de acuerdo con la Figura 2 se realiza la instalación y configuración del IDE IntelliJ IDEA, OpenJDK, Maven, Apache Tomcat y GitHub y posteriormente cargar un sistema web en un repositorio de GitHub, para luego ser montado, modificado y actualizado directamente desde el IDE IntelliJ. El entorno de trabajo utilizado, optimiza el proceso de desarrollo de sistemas y hace más eficaz el trabajo de los miembros del equipo distribuido.

Palabras clave: Equipo Distribuido – IntelliJ IDEA – OpenJDK – Maven – Tomcat – Repositorio – GitHub**.**

ABSTRACT:

The development of computer projects in distributed development teams was not possible today, thanks to tools that allow the proper development so that they are completed on time. Currently there are many development tools that facilitate this work, however due to the emergence of new technologies, many have not yet been explored. This research work focuses on the configuration of a work environment for the control of changes in distributed development team projects. The methodology of this work is experimental, according to Figure 2, the installation and configuration of the IDE IntelliJ IDEA, OpenJDK, Maven, Apache Tomcat and GitHub is carried out and then load a web system in a GitHub repository, to be later mounted, modified and updated directly from the IntelliJ IDE. The working environment used optimizes the system development process and makes the work of the members of the distributed team more efficient.

Keywords: Distributed Team - IntelliJ IDEA - OpenJDK - Maven - Tomcat - Repository - GitHub.

Introducción

El Trabajo Colaborativo en instituciones públicas y privadas, es una de las estrategias más útiles para agilizar la terminación de proyectos, el modelo de gestión de estos trabajos desde la perspectiva de monitoreo y evaluación es muy abierto y no está normado o estandarizado. La falta de repositorios de información que permitan evidenciar los aportes individuales de cada integrante del grupo, puede afectar en la no terminación de los proyectos en los que el equipo este trabajando. A partir de esta problemática, es que desde ya varios años, se ha venido integrando softwares de control de versiones que permiten a los equipos de desarrollo trabajar en un proyecto sin perder horas en papeleo y documentación, si no que estos softwares manejan un repositorio central organizado y lógico. En este contexto, ya queda a interpretación y al tipo de proyecto realizado, la selección de una de estas herramientas, para el equipo de desarrollo. Para elegir el control de versiones adecuado de un proyecto, se debe poner a consideración ventajas de los paquetes integrados, velocidad, funcionalidad, la cura de aprendizaje o capacidades de complementos IDE (Rawson, 2019). Para sustentar una herramienta de control de versiones en específico, es necesario evaluar los pro y contras de otras herramientas con el mismo objetivo, por lo cual se presenta un estudio comparativo de los principales sistemas de control de versiones en el presente año. Las herramientas evaluadas son: CVS, SVN, Mercurial y Git/GitHub.

Tabla 1. Sistemas de Control de versiones más populares - Ventajas y Desventajas
Fuente: (Rawson, 2019)

Sistema	Ventajas	Desventajas
CVS - Sistema de versiones concurrentes	• Vigente durante mucho tiempo por lo que se considera una tecnología madura.	• No se puede mover ni renombrar archivos en una nueva actualización. • Existen riesgos de seguridad de enlaces simbólicos a archivos. • No existe soporte para operaciones atómicas. • No esta diseñado para trabajos distribuidos.
SVN - Apache Subversión	• Mejoras basadas de CVS • Soporta operaciones atómicas. • Operaciones distribuidas. • Amplia gama de plugins para IDEs • No utiliza modelo de igual a igual	• Aún existen errores cuando se renombra archivos. • Insuficientes comandos de administración de repositorios • Es lenta en comparación a otras herramientas.
Git/GitHub	• OpenSource • Trabaja íntegramente con muchos IDEs • Incremento de velocidad de operación. • Operaciones de sucursales baratas (Cheap Branch operatios) • Árbol de historial completo disponible sin conexión. • Modelo distribuido de igual a igual.	• Es muy difícil de aprender para quienes trabajan con SVN • No esta echo para desarrolladores individuales. • Soporte limitado para Windows
Mercurial	• Más sencillo de aprender que Git • Tiene mejor documentación • Maneja un modelo distribuido	• Basado en extensiones en lugar de scriptability • Menos potencia fuera de la caja

A partir la información obtenida de la

Tabla *1*, se puede identificar algunos criterios importantes para nuestra investigación, los cuales permiten seleccionar la herramienta de control de versiones utilizada en el presente trabajo de investigación.

Criterios	**CVS**	**SVN**	**Git/GitHub**	**Mercurial**
Tecnología madura	X			
Fácil modificación de archivos			X	X
Soporta Operaciones atómicas		X	X	X
Operaciones distribuidas (sucursales)		X	X	
Velocidad Incremental			X	X
Soporte para trabajar con IDEs		X	X	X
Facilidad de uso			X	X
Accesibilidad al árbol de historial			X	
Modelo distribuido			X	X
Costo bajo	X		X	X
TOTAL	2	3	9	7

Tabla 2. Evaluación de los criterios de selección de herramienta de control de versiones.
Fuente: (Los autores)

De acuerdo con la **Tabla *2***, se identificaron un total de diez criterios, donde se consideró el desarrollo de software distribuido, en la cual la herramienta GitHub tiene una clara ventaja en cuanto a velocidad y distribución frente a sus competidores, con aproximadamente el 90% de aceptabilidad de estos criterios (**Figura 24**), es muy adecuado para proyectos que se prestan a sistemas distribuidos. Es así que el presente trabajo de investigación tiene como premisa el uso del sistema de control de cambios y versiones GitHub como herramienta de monitoreo para trabajos colaborativos, pues GitHub permite la creación ilimitada de repositorios, en los cuales se pueden alojar los trabajos colaborativos que el líder del equipo de desarrollo planifique y que los integrantes del equipo de trabajo puedan aportar a estos proyectos. Por defecto, los repositorios de GitHub son visibles para todos. Muchos proyectos deciden compartir su trabajo pública y abiertamente desde el inicio del proyecto para atraer la visibilidad y beneficiarse de las contribuciones de la comunidad desde el principio. Algunos otros grupos prefieren trabajar en privado en proyectos hasta que estén listos para compartir su trabajo. Los repositorios privados aseguran que el trabajo esté oculto, pero también limitan las colaboraciones a solo aquellos usuarios que tienen acceso al repositorio. Estos repositorios se pueden hacer públicos en una etapa posterior, como, por ejemplo, al enviar, aceptar o publicar los artículos de revistas correspondientes. En algunos casos, cuando la colaboración estaba destinada exclusivamente a ser privada, algunos repositorios nunca podrían hacerse públicos (Perez-Riverol et al., 2016).

En pocas palabras, la inicialización de un repositorio (local) (a menudo abreviado como repo) marca un directorio como uno a ser rastreado (Figura. 1). Todo o parte de su contenido se puede agregar explícitamente a la lista de archivos para rastrear.

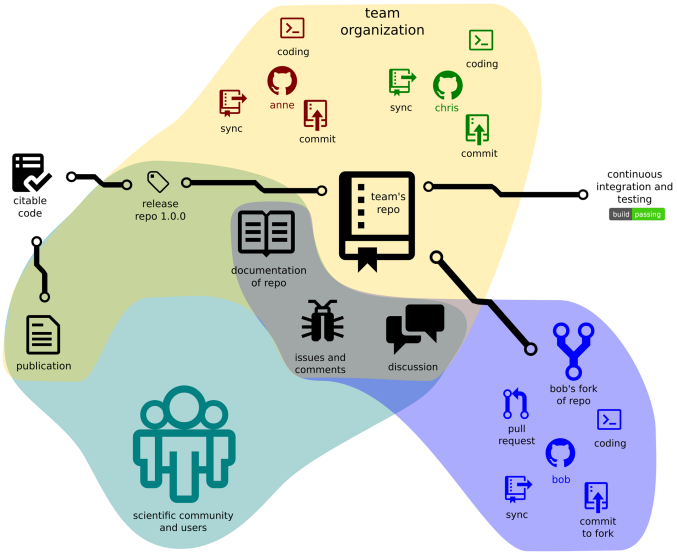

Figura 1. Estructura de un proyecto basado en GitHub que ilustra la estructura del proyecto y las interacciones con la comunidad.
Fuente: (Perez-Riverol et al., 2016)

Es cierto que existen herramientas de control de versiones para los repositorios de GitHub, sin embargo, en la actualidad los profesionales trabajan en equipos distribuidos por lo cual trabajan con nuevas tecnologías que brinden características para estos equipos de desarrollo, un ejemplar de estas tecnologías es el entorno de desarrollo IntelliJ IDEA (JetBrains, 2019), el cual es el IDE más completo que existe para crear un entorno apropiado en el que todos los miembros del equipo pueden trabajar juntos de manera eficiente. Integración transparente con una extensa variedad de sistemas de control de versiones facilita a los miembros del equipo permanecer en sincronía con los cambios de otros, certificando que todas las contribuciones serán fructíferas. IntelliJ IDEA puede coexistir con otras IDEs populares, como Eclipse y herramientas de gestión de proyectos como Maven, para que el equipo pueda utilizar cada herramienta donde sea mejor adaptable (Targetware, 2019). IntelliJ presenta dos distribuciones, una gratuita y de código abierto denominada **IntelliJ IDEA Community Edition** y otra de pago denominada **IntelliJ IDEA Ultimate**. Ambas son multiplataformas y están disponibles tanto para Windows como para Linux y Mac. IntelliJ presenta características que son muy interesantes, que difieren un poco de acuerdo con cada distribución, en este caso, la **Tabla 1.** se presenta las características de la versión gratuita. No obstante, dentro del apartado de licencias, que la versión de pago Ultimate la tendremos disponible, a modo de prueba, durante 30 días, para evaluar el rendimiento total del entorno de desarrollo.

Tabla 3. Características más importantes de IntellJ Community Edition
Fuente:(Academia Android, n.d.)

Análisis de Código	El editor enfatiza advertencias y errores inmediatamente facilitando aplicar una solución más rápida. Permite programación con intenciones y soluciones rápidas, soporte de controles Android Lint, perfiles de inspección compartidos.
Potente Editor	Permite la terminación de código inteligente y autocompletado más sofisticado y sensitivo. Recursos de previsualización y refactorizaciones avanzadas y seguras.
Herramientas integradas de Android	Posee un diseñador de interfaz de usuario que permite manipular elementos, apoyo a distintos esbozos y tipos de pantalla.
Aumento de Productividad	Brinda soporte para Maven y Gradle. Posee herramientas integradas para pruebas unitarias y de cobertura.
Lenguajes Soportados	Trabaja con varios lenguajes basados en JVM populares como son: Java, Scala, Groovy, Clojure y Kotin.

La versión de pago agrega una serie de características entre las que se puede destacar un mayor soporte a lenguajes, entre los que destacan:

- PHP, Python y Rubí.
- SQL, incluyendo PostgreSQL, MySQL, Oracle, SQL Server, etc…Además añade soporte avanzado para los marcos y estándares web más importantes.
- Desarrollo fácilmente con Spring MVC, Webflow, Jugar, Grails, Servicios Web, JSF, Struts, Flex y otros marcos.
- Incluye asistencia de código final para HTML5, CSS3, SASS, MENOS, Javascript, CoffeScript, Node js, ActionScript y otros lenguajes.

Por esta breve síntesis de lo que ofrece IntelliJ, se puede decir que se trata del IDE Java más inteligente que se puede hallar en el mercado. Por estar razones no es de extrañar que la propia Google haya usado de base IntelliJ para su IDE de desarrollo Android Studio. Como ya se había mencionado previamente, IntelliJ también agrega soporte para Maven, una herramienta open-source, que se creó en 2001 con el objetivo de simplificar los procesos de build que se tratan de los procesos de compilar y generar ejecutables a partir del código fuente (Ana M. Del Carmen García Oterino, 2014).

Maven es una herramienta capaz de gestionar un proyecto software completo, desde la etapa en la que se comprueba que el código es correcto, hasta que se despliega la aplicación, pasando por la ejecución de pruebas y generación de informes y documentación (Ana M. Del Carmen García Oterino, 2014). Antes de que Maven proporcionara una interfaz común para hacer builds del software, cada

proyecto solía tener a alguna persona dedicada exclusivamente a configurar el proceso de build, por eso es que Maven simplifica mucho el proceso de build del código, permitiéndonos compilar cualquier tipo de proyecto de la misma manera, librándonos de todas las dificultades que hay por detrás. Como lo muestra la **Tabla 4** , Maven establece tres ciclos de build del software con una sucesión de etapas particulares.

Tabla 4. Maven - Ciclos de build del software.
Fuente: (García, 2014).

Validación	Aprobar que el proyecto es apropiado.
Compilación	Compilar el proyecto.
Pruebas	Someter a pruebas el código fuente usando un framework de pruebas individuales.
Empaquetar	Compactar el código compilado y convertirlo al formato .jar o .war
Pruebas de Integración	Procesar y desplegar el código en cierto entorno en el que se logren ejecutar las pruebas de integración.
Verificación	Comprobar que el código empaquetado es legítimo y cumple los criterios de calidad.
Instalación y Despegue	Instalar y desplegar el código empaquetado en el repositorio local de Maven, para emplearlo como dependencia de otros proyectos.

Para poder llevar a cabo alguna de estas fases en nuestro código, tan solo se debe ejecutar mvn y el nombre de la fase (la palabra que puse entre paréntesis). Además, van en cadena, es decir, si empaquetamos el código (package), Maven ejecutará desde la fase de validación (validate) a empaquetación (package).
IntelliJ también soporta servidores como Apache Tomcat, una implementación ampliamente utilizada de la especificación Java Servlet, que ha sido desarrollada como un proyecto de código abierto por la Apache Software Foundation desde 1999, cuando la fuente del proyecto fue donada a ASF por Sun Microsy (MuleSoft, n.d.).

El servidor Apache Tomcat es un contenedor de aplicaciones web basadas en Java de código abierto que se creó para ejecutar servlet y aplicaciones web JavaServer Pages (JSP). Fue creado bajo el subproyecto Apache-Jakarta; sin embargo, debido a su popularidad, ahora se aloja como un proyecto de Apache separado, donde es apoyado y mejorado por un grupo de voluntarios de la comunidad Java de código abierto (Thomas & Engineer, 2010). Apache Tomcat es muy estable y tiene todas las características de un contenedor de aplicaciones web comerciales, pero está sujeto a la Licencia de Apache de Código Abierto. Tomcat también proporciona una funcionalidad adicional que lo convierte en una excelente opción para desarrollar una solución de aplicación web completa. Algunas de las funciones adicionales proporcionadas por Tomcat, además de ser de código abierto y gratuitas, incluyen la aplicación Tomcat Manager, las implementaciones especializadas en el ámbito y las válvulas Tomcat. Las versiones actualmente admitidas en Apache Tomcat son 5.5X, 6.0X y 7.0X. Las versiones anteriores a la

5.5 aún están disponibles para descargar, pero están archivadas y no hay soporte disponible para ellas, por lo que se recomienda a los usuarios que usen la última versión posible de Tomcat cuando esté disponible. Versiones principales en Apache Tomcat coinciden con versiones de la especificación de Java Servlet, o Java Servlet API, lanzadas. Por lo tanto, Tomcat 5.5X admite Servlet API 2.3, Tomcat 6.0X admite Servlet API 2.4 y la última versión de Tomcat 7.0 es una implementación de referencia de la actual Servlet API 3.0. Además de las versiones de la API de Servlet, las versiones de Tomcat admiten las correspondientes versiones de la API de JSP.

En resumen:

IntelliJ es un entorno de desarrollo integrado, el usuario puede integrar Maven y Tomcat en esta herramienta de desarrollo si es necesario. Se puede usar Maven para administrar su proyecto y las dependencias de su proyecto, se descargará desde los repositorios centrales u otros repositorios remotos a sus repositorios locales de acuerdo con las configuraciones que establezca en pom.xml. GitHub es el sistema de control de versiones distribuido más avanzado.
.

Materiales y Métodos

Las herramientas utilizadas para la configuración del entorno de desarrollo, del presente trabajo de investigación, son las siguientes:

- Hardware: Laptop HP Windows 10 Home 17’ (i7-7500U CPU @ 2.70GHz 2.90 GHz).
- Sistema Operativo: CentOS 7
- Entorno de desarrollo
    - ✓ OpenJDK 8 JDK
    - ✓ IntelliJ IDEA Ultimate (v. 2018.3.4).
    - ✓ Apache Tomcat 7.0.76.
    - ✓ Apache Maven 3.
    - ✓ GitHub.

Por otro lado, la metodología utilizada se describe en la **Figura 2**, donde se muestra en manera resumida y ordenada las etapas de exploración del uso de herramientas como GitHub e IntelliJ para el desarrollo de sistemas en equipos distribuidos.

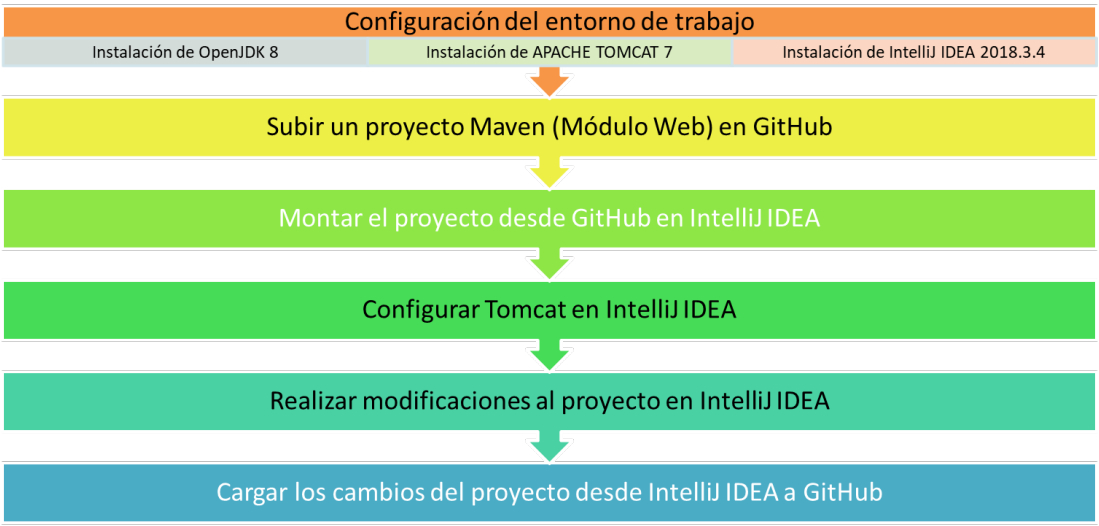

Figura 2. Metodología utilizada en la Investigación.
Fuente: Los Autores

A continuación, se detalla cada una de las etapas de la metodología utilizada, detallando el resultado después de cada etapa.

Configuración del Entorno.

- **Instalación de OpenJDK 8**

Para instalar OpenJDK 8 JDK usando yum, ejecute este comando:

```
sudo yum install java-1.8.0-openjdk-devel
```

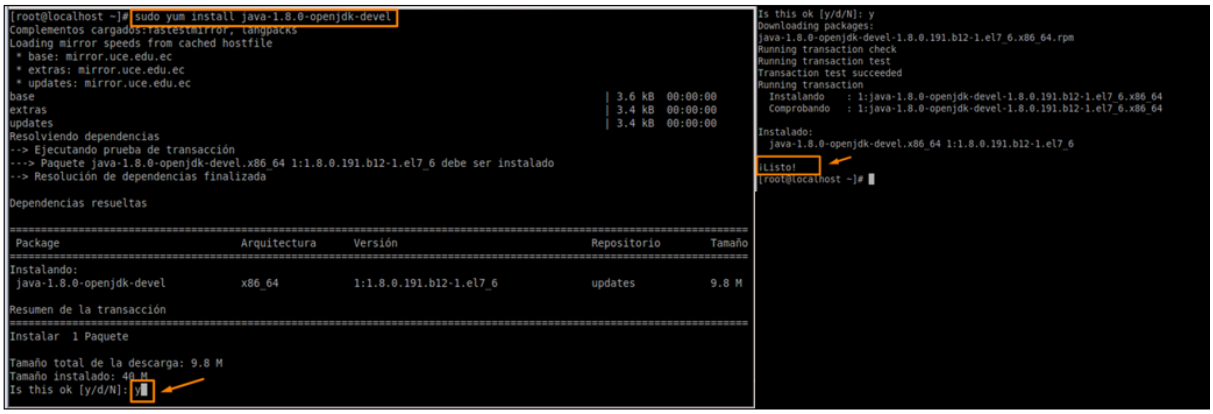

Figura 3. Instalación de OpenJDK 8

La **Figura *3*** muestra la instalación de OpenJDK mediante el comando ***yum***, desde la terminal de Linux.

- **Instalación de APACHE TOMCAT 7**

Desde el administrador (root), se escribe en consola: `yum install tomcat`. Al finalizar aparecerá un mensaje indicando que la instalación está terminada (**Figura 4**).

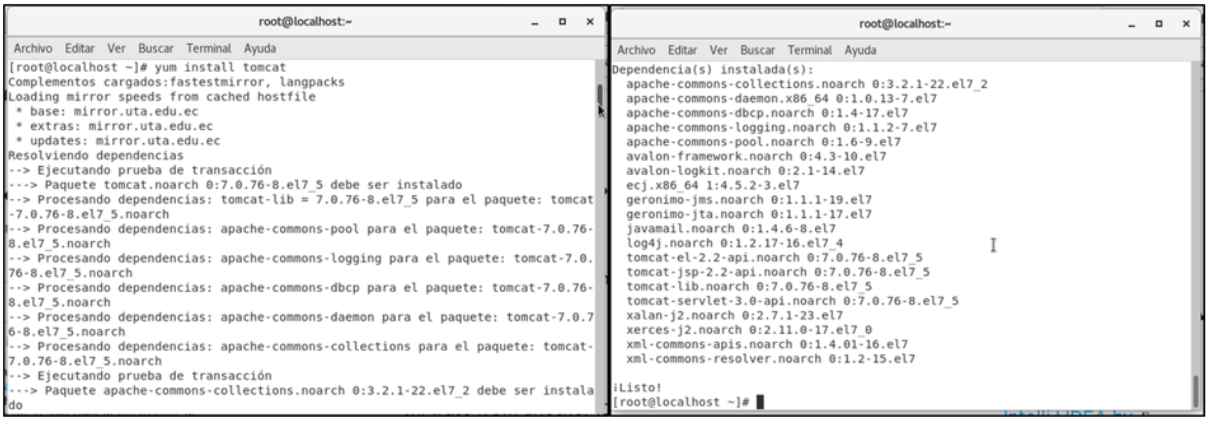

Figura 4. Instalación de Apache Tomcat - Linux

Posteriormente se escribe **`gedit /usr/share/tomcat/conf/tomcat.conf`** y se agrega la línea descrita en la **Figura *5***, al archivo **tomcat.conf**. Se guardan los cambios y se cierra el editor de líneas.

```
#JAVA_OPTS adicional
JAVA_OPTS="-Djava.security.egd=file:/dev/./urandom -Djava.awt.headless=true -Xmx512m -XX:MaxPermSize=256m -XX:+UseConcMarkSweepGC"
```

Figura 5. Configuración del archivo tomcat.conf

Luego se escribe **`yum install tomcat-webapps`** para instalar los paquetes de administración y se confirma la instalación (**Figura *6***).

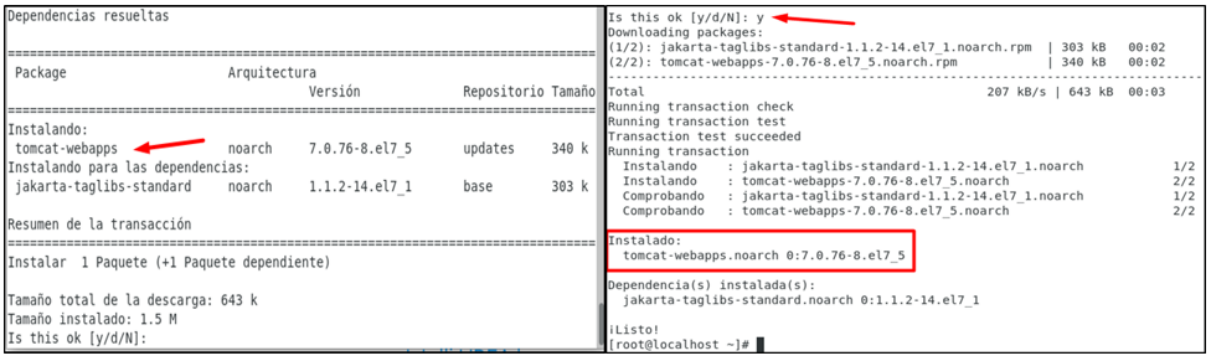

Figura 6. Instalación de los paquetes de administración de tomcat - Linux

Posteriormente se modifica el archivo **tomcat-users.xml** . Desde el usuario root se escribe: **`gedit /usr/share/tomcat/conf/tomcat-user.xml`** y se agrega debajo de la línea **<tomcat-users>** las dos líneas siguientes:

```
<user username="admin" password="password" roles="manager-gui,admin-gui"/>
</tomcat-users>
```

Finalmente se inicia el servicio de Apache Tomcat, mediante el comando:

`systemctl start tomcat`

Para comprobar si el servicio de Apache Tomcat, funciona correctamente se debe dirigir a la dirección local y puerto de escucha, es decir desde un navegador escribir: `localhost:8080.` (**Figura 7**)

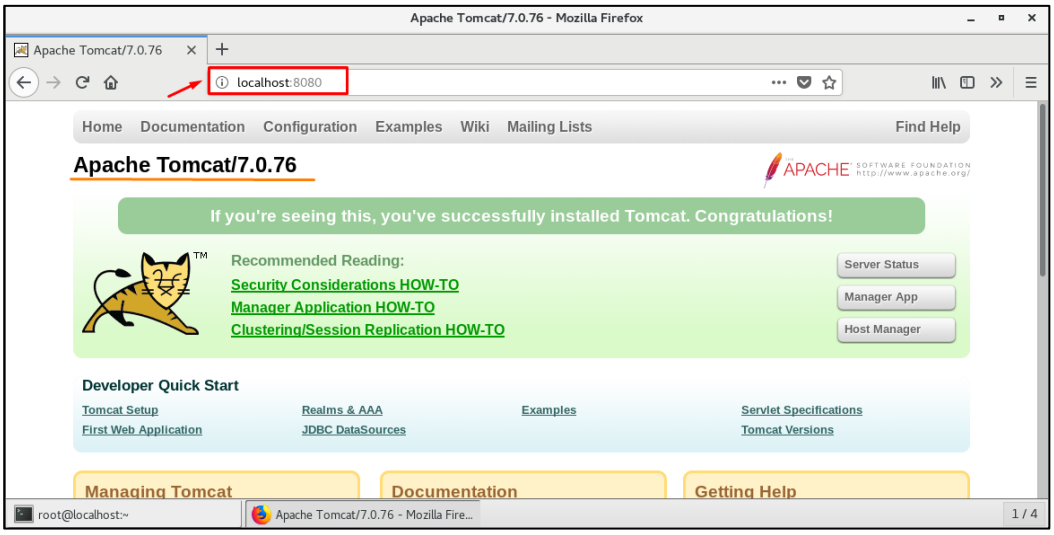

Figura 7. Servicio de Apache Tomcat Levantado

Como se aprecia en la **Figura 7**, se carga el servidor de Tomcat correctamente, y en la parte derecha se puede acceder a las opciones para administrar el servidor.

Instalación de IntelliJ IDEA 2018.3.4 (“Install and set up IntelliJ IDEA,” 2018)

Para instalar IntelliJ IDEA, se debe seguir los siguientes pasos:

a. Ir a la página principal de IntelliJ IDEA. (Jetbrains, 2019)
b. Seleccionar el Sistema Operativo y distribución que se vaya a usar (**Figura 8**).

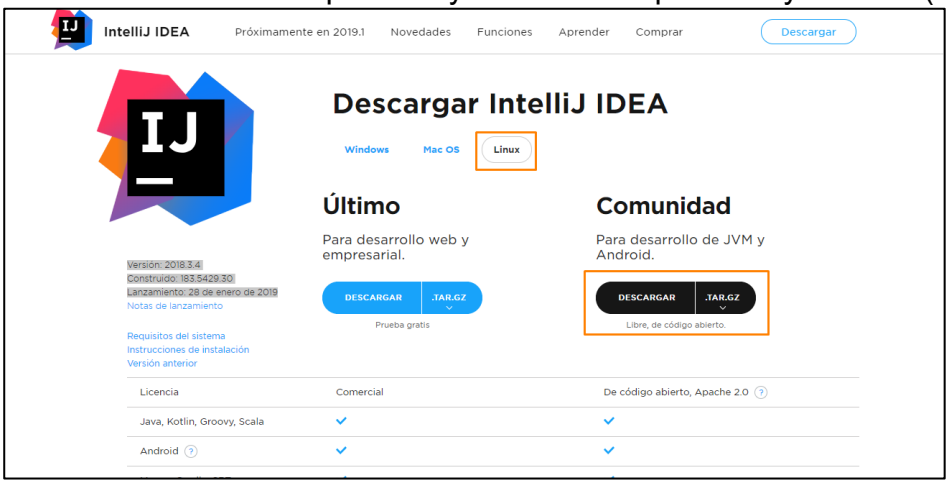

Figura 8. Página de Descarga de IntelliJ IDEA

c. Mover el archivo **idealC.gz** o **idealU.gz** que ha descargado en una carpeta diferente si la carpeta de Descargas actual no permite la ejecución de archivos

d. La ubicación de instalación recomendada según el estándar de jerarquía del sistema de archivos (FHS) es /opt. Se puede Ingresar el siguiente comando:

`mv ideaIU-2018.3.4.tar.gz ../../opt/`

e. Descomprimir el archivo **idealC.gz** o **idealU.gz** que ha descargado y desplazado al directorio actual (**Figura *9***). Colocar el siguiente comando:

`tar xfzv ideaIU-2018.3.4.tar.gz`

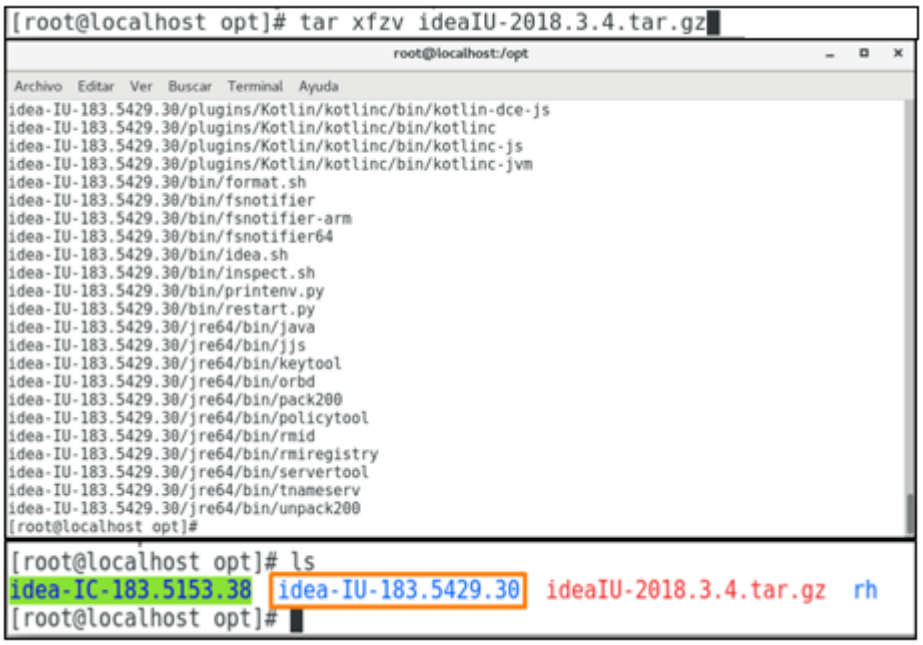

Figura 9. Descomprensión del archivo idealU.gz

f. Dirigirse al directorio **`idea-IU-183.5429.30/bin/`**

g. Ejecutar el **idea.sh** desde el subdirectorio **`idea-IU-183.5429.30/bin/`** (**Figura *10***)

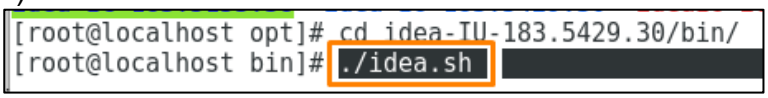

Figura 10. Comando de Instalación - IntelliJ IDEA

Posteriormente se inicializa el entorno de desarrollo IntelliJ IDEA 2018.3.4 (**Figura *11***)

Figura 11. Entorno de instalación - IntelliJ IDEA 2018.3.4

Para probar y entender cómo se maneja un control de versiones con las herramientas instaladas y configuradas anteriormente, se realizan las siguientes actividades descritas en la **Figura 2**, como parte de la metodología usada en la presente investigación:

a. Subir un proyecto Maven (Módulo Web) en GitHub y copiar su dirección web.
b. Montar el proyecto con IntelliJ IDEA haciendo uso de Git
c. Asegurarse que IntelliJ entienda que se trata de un proyecto Maven
d. Configurar Tomcat en el IDE para que se ejecute el proyecto.
e. Realizar cambios en el proyecto: añadir dependencias y crear nuevas clases y ficheros.
f. Cargar los cambios al repositorio directamente desde IntelliJ

a. Subir un proyecto Maven a IntelliJ desde GitHub.

IntelliJ trabaja con GitHub a través de Git, el que permite cargar proyectos desde la web. Inicialmente se tiene un proyecto en Maven (aplicación web) alojado en el repositorio principal del jefe del proyecto (Wilson Steeven, 2019). Al iniciar la aplicación IntelliJ, se escoge la opción de Git, la misma que se encuentra en la lista desplegable de la opción “Check out from version control “(**Figura 12**).

.

Figura 12. Ventana principal de IntelliJ IDEA

Posteriormente se ingresa la dirección url del repositorio en el que se encuentra la aplicación web y el directorio donde está asignada la descarga de la copia local del proyecto (**Figura 13**). Así mismo, se realiza algunas configuraciones para que el IDE IntelliJ IDEA, reconozca al aplicativo web como un proyecto Maven.

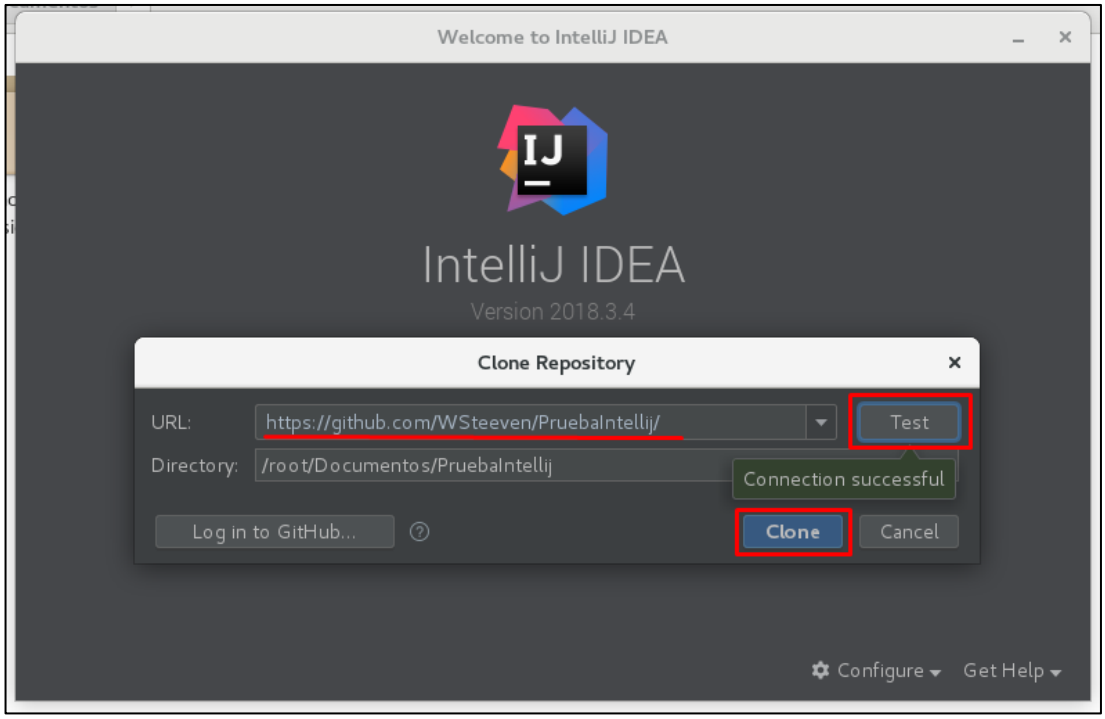

Figura 13. Clone Repository - IntelliJ IDEA

Se realizan las configuraciones necesarias para poder cargar el proyecto, y que este sea reconocido como un proyecto de Maven, y posteriormente selecciona la versión de java instalada con anterioridad (**Figura 14**).

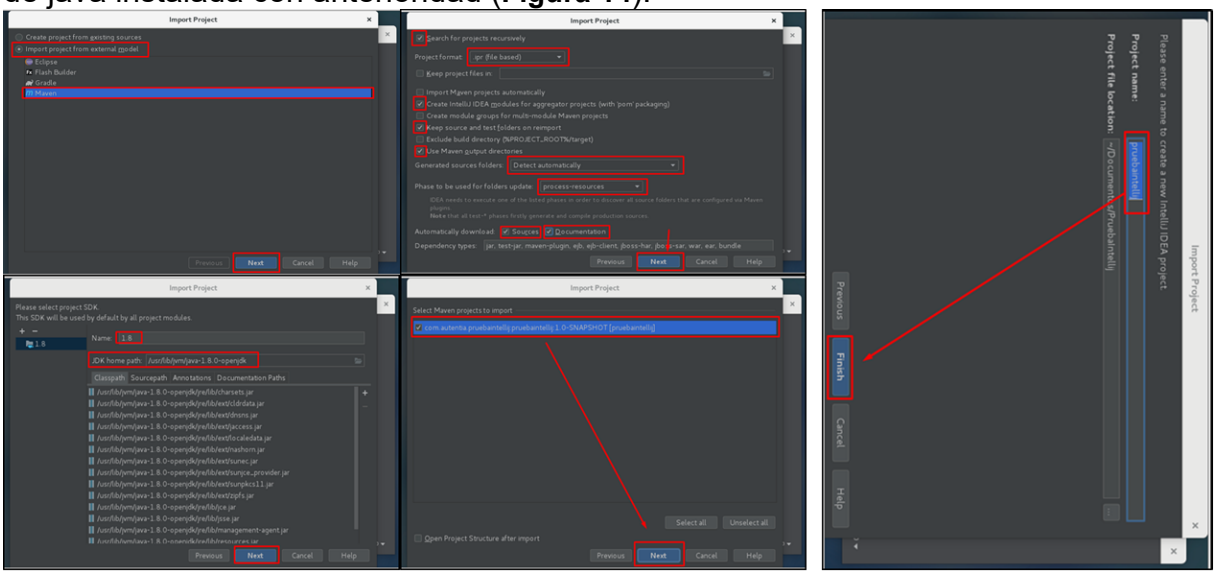

Figura 14. Configuraciones del proyecto en IntelliJ IDEA

La **Figura 15** muestra el entorno de trabajo del IDE IntelliJ IDEA, con el proyecto en Maven ya cargado y listo para trabajar en las modificaciones.

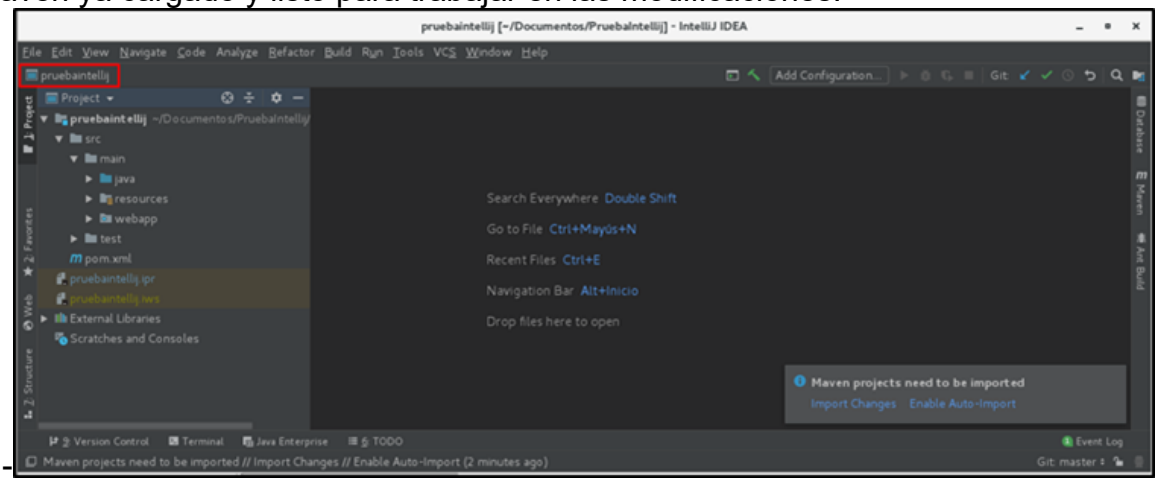

Figura 15. Entorno de trabajo de IntelliJ IDEA

b. Configuración de Apache Tomcat 7

La configuración del servidor de Apache Tomcat desde IntelliJ se la realiza mediante la opción “Add Configuration” y luego seleccionar la opción de Tomcat server de manera local (**Figura *16***).

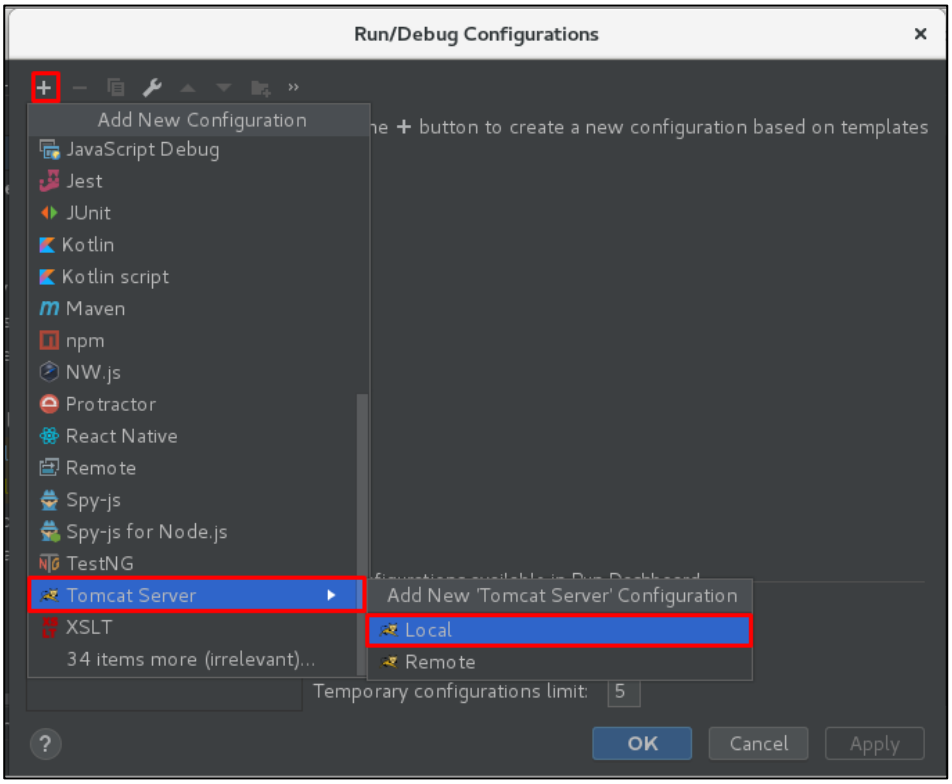

Figura 16. Run/Debug Configurations - IntelliJ IDEA

Posteriormente se agregan los parámetros convenientes para que el servidor Tomcat pueda trabajar sin ningún inconveniente. En la pestaña "Deployment" se selecciona la aplicación que va a correr en el servidor (**Figura 17**).

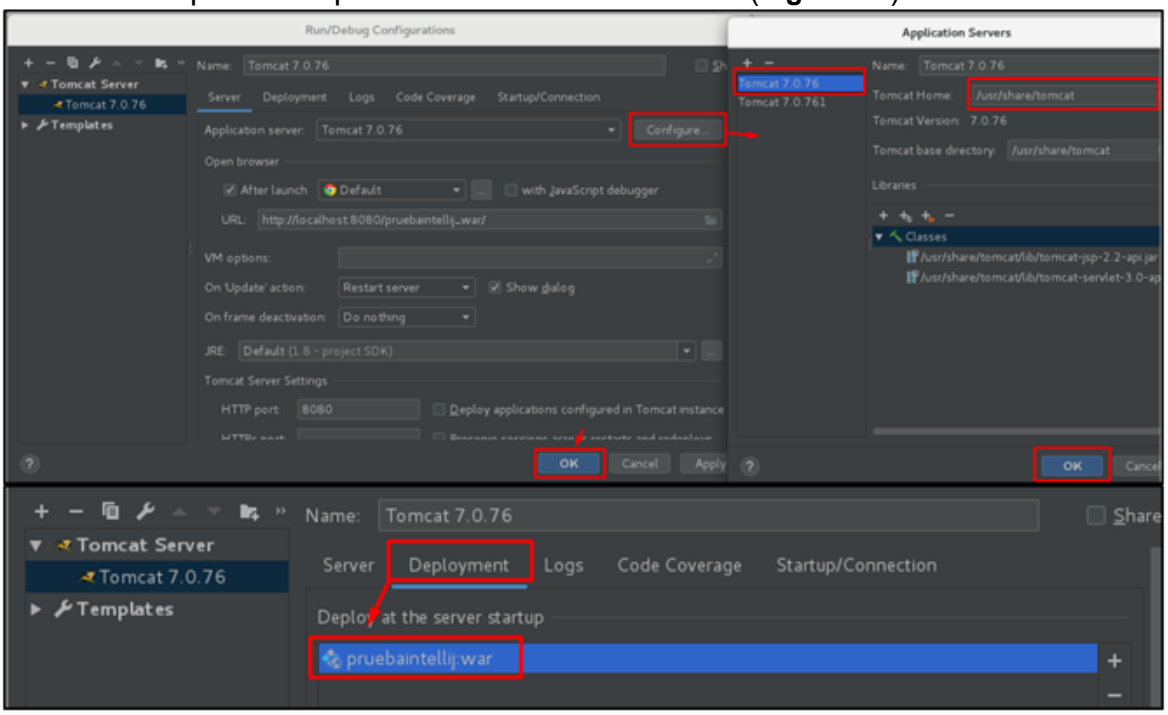

Figura 17. Configuración de Apache Tomcat desde ItelliJ IDEA

La **Figura 18** muestra al servidor de Apache Tomcat configurado y listo para la implementación del proyecto cargado anteriormente, el cual se aprecia con su nombre y una extensión de archivo **.war** lo que indica que se puede ejecutar la aplicación desde el servidor de apache Tomcat.

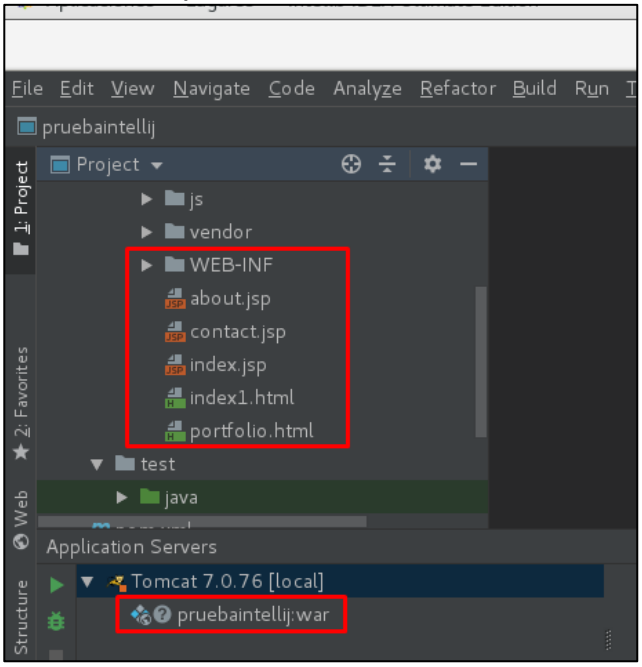

Figura 18. Apache Tomcat configurado desde IntelliJ IDEA

Si se ejecuta el build de la aplicación, el servidor Tomcat arranca y despliega correctamente la aplicación web (**Figura 19**).

Figura 19. Aplicación Web ejecutada localmente desde IntelliJ IDEA mediante Apache Tomcat

c. Modificando el Proyecto

Cuando se tiene ya el proyecto de Maven cargado en IntelliJ IDEA, solo queda su respectiva modificación de los archivos principales de la aplicación web, por ejemplo se podría cambiar el título de cabecera “Juan Pérez” por el nombre de los integrantes del equipo, responsables del presente trabajo de investigación (**Figura 20**).

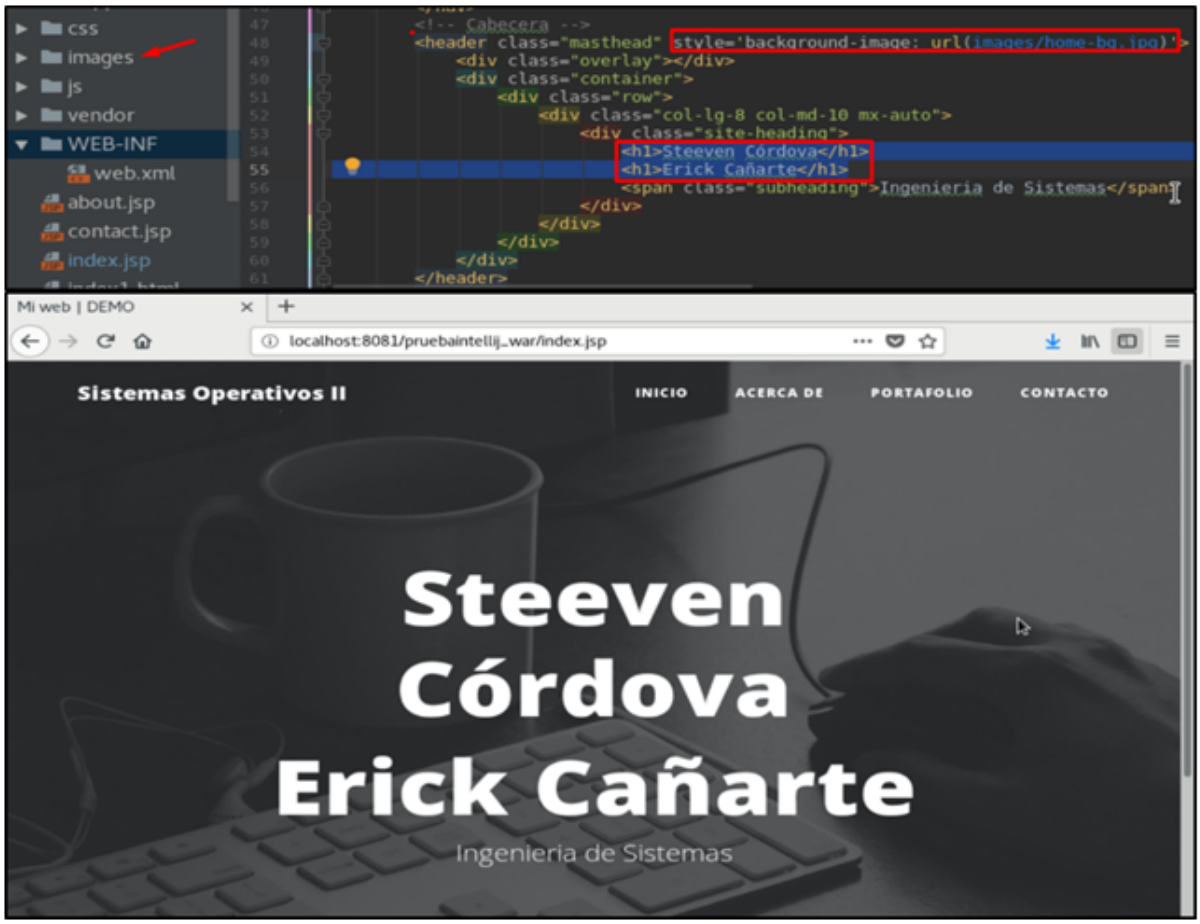

Figura 20. Aplicación Web modificada localmente desde IntelliJ IDEA

d. Subiendo los cambios al repositorio.

Se ingresa las credenciales de usuario de GitHub, para luego confirmar la realización de los los cambios. Luego se presiona en **Commit and Push** y nuevamente se ingresa las credenciales de la cuenta de GitHub para iniciar sesión en el repositorio desde IntelliJ IDEA (**Figura 21**). Los cambios se realizan en el repositorio mostrándose un mensaje de confirmación de actualización del proyecto (**Figura 22**).

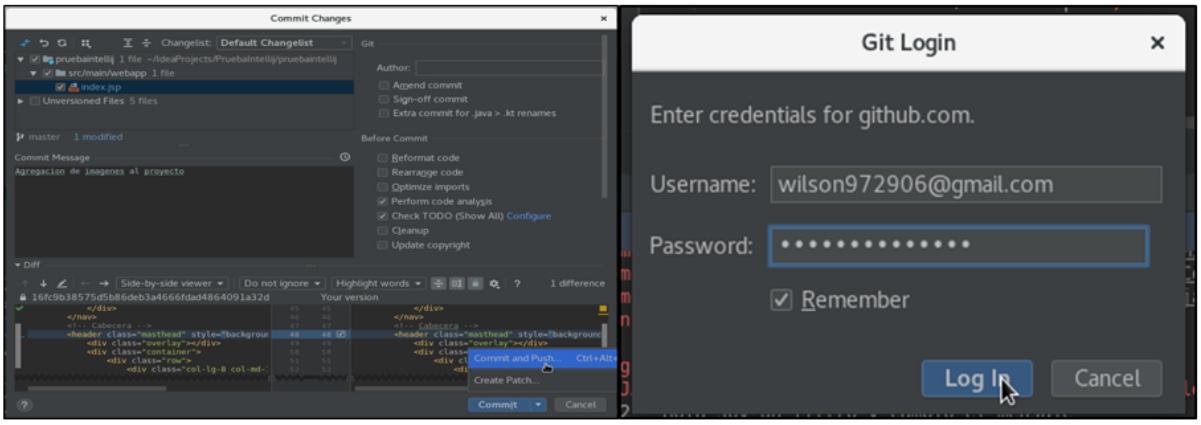

Figura 21. Proceso de Commit and Push

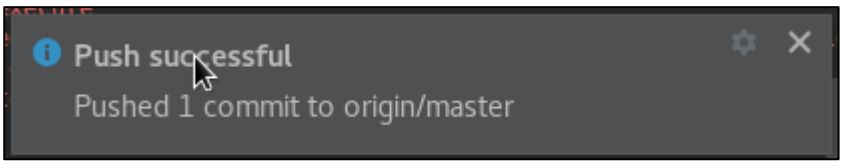

Figura 22. Mensaje de confirmación de actualización del proyecto en GitHub

La **Figura 23** refleja el registro registrado de los cambios cargados al repositorio de GitHub con las descripciones que se indicaron anteriormente en la opción de **push and commit.**

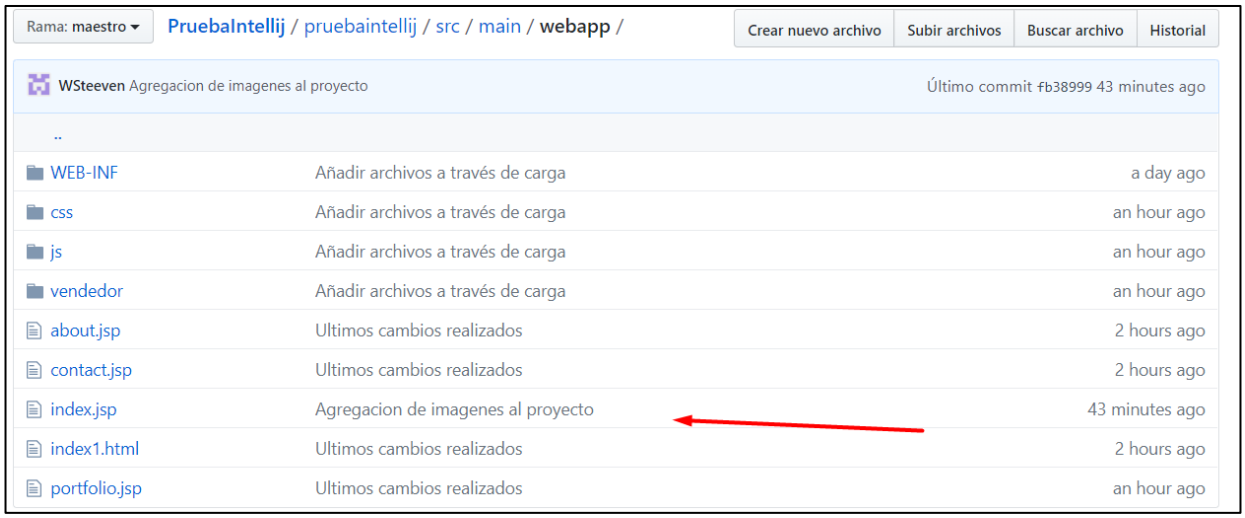

Figura 23. Cambios actualizados en el repositorio principal del proyecto (Aplicación Web)

Discusión de Resultados

Los resultados obtenidos durante el presente trabajo de investigación son de carácter práctico-experimental, aunque también se hicieron hallazgos teóricos que se trae a citación por justificación de la herramienta de control de versiones usada en esta investigación. La **Figura 24** *es resultado de la avaluación de los criterios de selección identificados en la* **Tabla 2**, *en esta se proponen una serie de diez criterios a evaluar, donde se marcó subjetivamente los criterios de selección para equipos*

distribuidos que mantienen cada herramienta de control de versiones evaluada, gracias a la información de la

Tabla *1*, obteniendo como resultado un 20% de CVS,30% de SVN, 90% de Git/GitHub y un 70% de Mercurial aproximadamente.

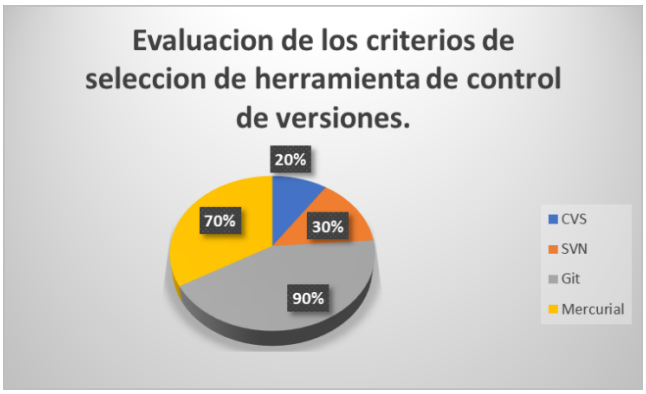

Figura 24. Evaluación de los criterios de selección de herramienta de control de versiones
Fuente: (Los autores)

De acuerdo con Rawson, "si se está encabezando un proyecto de código abierto en el que varios programadores trabajarán en diferentes instantes y/o enviarán varias actualizaciones al código, Git/GitHub es una excelente opción para su proyecto debido al enorme aumento de la velocidad y la mejora de la gestión de árboles de historial"(Rawson, 2019). Este contexto se evidencia al seguir la metodología secuencial descrita en la **Figura 2,** la cual establece seis etapas, descritas a continuación: Primera Etapa – Configuración del entorno de trabajo; Segunda Etapa – Subir un proyecto Maven en GitHub; Tercera Etapa – Montar el proyecto desde GitHub en IntelliJ IDEA; Cuarta Etapa – Configurar Tomcat en IntelliJ IDEA; Quinta Etapa – Realizar modificaciones al proyecto en IntelliJ IDEA; Sexta Etapa - Cargar los cambios del proyecto desde IntelliJ IDEA a GitHub

Dentro de la primera etapa, se encuentran las respectivas instalaciones de las herramientas utilizadas en el presente trabajo de investigación, de la instalación de OpenJDK 8 (**Figura *3***), Apache Tomcat (**Figura *4***) e IntelliJ IDEA (**Figura *11***). A partir de la segunda etapa ya es meramente el núcleo del trabajo, donde la **Figura *12*** representa el entorno principal del IntelliJ IDEA, con el proyecto en Maven ya cargado en el repositorio y abierto en este IDE. Posteriormente se realiza la configuración del servidor de Apache Tomcat tal y como se representa en la **Figura *17***, con las configuraciones anteriormente ejecutadas a la perfección, se puede trabajar en el programa que se está desarrollando, para luego realizar los respectivos cambios al repositorio GitHub, donde se encuentra alojado el sistema principal (**Figura *21***) luego posteriormente tal y como lo muestra la **Figura *23***, estos cambios se reflejan en el repositorio de GitHub.

Las impresiones al tratar con el entorno de trabajo evaluado son considerablemente positivas, Llama la atención herramientas como IntelliJ IDEA por la cantidad de tecnologías que soporta: Groovy, Android, Javascript, Struts, Spring, Hibernate, JSF, etc. Además integra funcionalidades para trabajar con herramientas de control de versiones, lo cual le da un valor agregado para la productividad dentro

de las empresas, principalmente aquellas empresas que tiene equipos de desarrollo distribuido.

Conclusiones

El sistema de control de versiones GitHub, es uno de los más populares en la actualidad, por sus características de velocidad, fácil acceso y uso, operaciones distribuidas, entre otras, se mantiene en un puesto de preferencia por los programadores y equipos de desarrollo distribuidos.

IntelliJ IDEA es uno de los IDE que venido ganando terreno en el desarrollo de sistemas en equipos distribuidos, ya que este brinda soporte para trabajar con tecnologías como Maven, Tomcat y GitHub, permitiéndole trabajar y tener un control de versiones de todos sus proyectos, haciendo de este el IDE “más inteligente”.

El entorno de desarrollo utilizado en el presente trabajo de investigación (IntelliJ IDEA, GitHub, OpenJDK, Apache Tomcat y Apache Maven), permitió experimentar el uso de una alternativa de buenas prácticas, para equipos de desarrollo distribuidos, optimizando los tiempos de producción de los proyectos informáticos en los que estos equipos se encuentren trabajando.

Referencias Bibliográficas

Academia Android. (n.d.). IDE para Android: IntelliJ IDEA, Android Studio y AIDE – Academia Android. Retrieved February 7, 2019, from https://academiaandroid.com/ide-android-intellij-android-studio-aide/

Ana M. Del Carmen García Oterino. (2014). Simple y rápido. Entiende qué es Maven en menos de 10 min. - Javier Garzás. Retrieved February 7, 2019, from https://www.javiergarzas.com/2014/06/maven-en-10-min.html

Install and set up IntelliJ IDEA. (2018). Retrieved February 7, 2019, from https://www.jetbrains.com/help/idea/install-and-set-up-product.html

Jetbrains. (2019). Descarga IntelliJ IDEA: el IDE de Java para desarrolladores profesionales de JetBrains. Retrieved February 7, 2019, from https://www.jetbrains.com/idea/download/index.html#section=linux

MuleSoft. (n.d.). An Introduction To Tomcat Catalina | MuleSoft. Retrieved February 7, 2019, from https://www.mulesoft.com/tcat/tomcat-catalina

Perez-Riverol, Y., Gatto, L., Wang, R., Sachsenberg, T., Uszkoreit, J., Leprevost, F. da V., … Vizcaíno, J. A. (2016). Ten Simple Rules for Taking Advantage of Git and GitHub. *PLoS Computational Biology*, *12*(7), 1–11. https://doi.org/10.1371/journal.pcbi.1004947

Rawson, R. (2019). 2019 Version Control Software Comparison: SVN, Git, Mercurial. Retrieved December 26, 2019, from https://biz30.timedoctor.com/git-mecurial-and-cvs-comparison-of-svn-software/

Thomas, M., & Engineer, S. S. (2010). Introduction to Apache Tomcat 7 . 0. *Agenda*, (August), 7–8.

Wilson Steeven. (2019). WSteeven / PruebaIntellij. Retrieved from https://github.com/WSteeven/PruebaIntellij/