ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

"IMPLEMENTACIÓN DE UN REPOSITORIO DE ARTEFACTOS DE SOFTWARE NEXUS, EN UN ENTORNO DE INTEGRACIÓN CONTINUA PARA LA DIRECCIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES DEL EJÉRCITO ECUATORIANO".

CUEVA CUEVA, FRANKLIN EFREN

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE TECNOLOGÍA EN COMPUTACIÓN

MONOGRAFÍA PREVIA A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN COMPUTACIÓN

ING. CAIZA CAIZABUANO, JOSÉ RUBÉN

LATACUNGA, 04 DE SEPTIEMBRE DE 2020

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE TECNOLOGÍA EN COMPUTACIÓN

CERTIFICACIÓN

Certifico que el trabajo de titulación, "IMPLEMENTACIÓN DE UN REPOSITORIO DE ARTEFACTOS DE SOFTWARE NEXUS, EN UN ENTORNO DE INTEGRACIÓN CONTINUA PARA LA DIRECCIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES DEL EJÉRCITO ECUATORIANO", fue realizado por el Señor Cueva Cueva, Franklin Efren, el mismo que ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con les requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustenten públicamente.

Latacunga, 04 de Septiembre de 2020

Ing. Caiza Caizabuano, José Rubén
Director del Proyecto Técnico

URKUND

Document Information

Analyzed document	MonografiaCuevaFranklinV2.docx (D78466103)
Submitted	9/1/2020 8:52:00 PM
Submitted by	
Submitter email	jrcaiza@espe.edu.ec
Similarity	5%
Analysis address	jrcaiza.espe@analysis.urkund.com

Sources included in the report

W	URL: https://www.javiergarzas.com/2014/08/nexus-artifactory-10-min.html Fetched: 9/1/2020 8:53:00 PM	12
W	URL: https://rdu.iua.edu.ar/bitstream/123456789/1781/1/TFG.pdf Fetched: 6/6/2020 6:01:50 AM	1
SA	**Franklin_Tandalla_Entrega_Recuperación.pdf** Document Franklin_Tandalla_Entrega_Recuperación.pdf (D49412435)	1
SA	**GMCA-DISEÑO DE LA PLATAFORMA DE LIBROS DIGITALES DE LA UCE.docx** Document GMCA-DISEÑO DE LA PLATAFORMA DE LIBROS DIGITALES DE LA UCE.docx (D63603610)	2
W	URL: https://docplayer.es/88347592-Departamento-de-ciencias-de-la-computacion.html Fetched: 12/20/2019 7:53:13 AM	2
W	URL: https://docplayer.es/amp/58876288-Sistema-de-gestion-de-pagos-electronicos-manteni ... Fetched: 12/14/2019 11:53:56 AM	2
W	URL: https://javiermartinalonso.github.io/drafts/DEV-OPS/vagrant/2018-12-31-nexus.html Fetched: 9/1/2020 8:53:00 PM	2
W	URL: https://blog.moon.cat/instalar-nexus-repository-manager-maven/ Fetched: 9/1/2020 8:53:00 PM	1

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE TECNOLOGÍA EN COMPUTACIÓN

RESPONSABILIDAD DE AUTORÍA

Yo, Cueva Cueva, Franklin Efren, declaro que contenido, ideas y criterios de este trabajo de titulación "IMPLEMENTACIÓN DE UN REPOSITORIO DE ARTEFACTOS DE SOFTWARE NEXUS, EN UN ENTORNO DE INTEGRACIÓN CONTINUA PARA LA DIRECCIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES DEL EJÉRCITO ECUATORIANO". Es de mi autoría y responsabilidad cumpliendo con los requisitos legales teorías científico, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Latacunga, 04 de septiembre de 2020

Cueva Cueva, Franklin Efren
C.C.: 1104807811

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELECTRONICA Y COMPUTACIÓN

CARRERA DE TECNOLOGÍA EN COMPUTACIÓN

AUTORIZACIÓN DE PUBLICACIÓN

Yo, Cueva Cueva, Franklin Efren, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el presente trabajo de titulación "IMPLEMENTACIÓN DE UN REPOSITORIO DE ARTEFACTOS DE SOFTWARE NEXUS, EN UN ENTORNO DE INTEGRACIÓN CONTINUA PARA LA DIRECCIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES DEL EJÉRCITO ECUATORIANO". En el repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Latacunga, 04 de septiembre de 2020

Cueva Cueva, Franklin Efren
C.C.: 1104807811

DEDICATORIA

El presente trabajo de titulación lo quiero dedicar primeramente a Dios por haberme dado salud y vida para poder llegar a concluir con éxito este trabajo y culminar está meta propuesta.

A mi familia y en especial a mi hija Thais quienes son el pilar y soporte fundamental de mi vida profesional, por darme las fuerzas para seguir adelante y ser mi motivación diaria para seguir en pie en lucha de mis ideales. Gracias por su paciencia y comprensión.

Finalmente, y de manera muy especial quiero dedicárselo a todas las personas que estuvieron conmigo en todo momento, ya sea físicamente o con una palabra de aliento, para no dejarme caer en este duro camino que me ha tocado recorrer y que me han inspirado para seguir con pie firme mi propósito.

Cueva Cueva, Franklin Efren

AGRADECIMIENTO

Infinitamente agradecido con Dios por todas sus bendiciones y permitirme llegar con éxito a la culminación de mi carrera, además por cada uno de los días que me brinda.

En este proyecto quiero reflejar mi sincero agradecimiento a mi familia y todas las personas que me brindaron su apoyo incondicional en esta etapa de mi vida, quienes hicieron que este camino fuese más llevadero con sus consejos y paciencia. Además, agradezco infinitamente a los señores docentes de la Unidad de Gestión de Tecnologías, que, durante este tiempo en las aulas, impartieron sus conocimientos y consejos en bien de cada uno de nosotros, en especial a mi director del proyecto José Caiza, por guiarme y ayudarme con su asesoría de una manera muy profesional en la realización de este proyecto.

Cueva Cueva, Franklin Efren

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

RESUMEN

La DTIC, es la entidad encargada de planificar, diseñar y ejecutar proyectos de desarrollo, donde cada desarrollador trabaja independientemente con un repositorio local conocido como Maven y un repositorio remoto y publico denominado Maven Central, al copilar el código Maven buscara en el repositorio local si están instaladas las librerías y si no están irá a buscarlas a Maven Central, las descargara e instalará en nuestro repositorio local, la próxima vez que Maven copile hará el mismo proceso, pero como ya están descargadas será más rápido y no le hará falta descargarlas.

Es aquí donde interviene el Administrador de Repositorios de Nexus, como FF. AA no siempre nos interesa depender da algo externo a la organización y siempre poder mantener controladas las librerías propias ya que son de carácter confidencial y reservado. La idea principal de trabajar con Nexus es que sea un repositorio de artefactos de software internos de la organización y sea compartido por todo el equipo de desarrollo. Al tener instalado Nexus al momento de copilar o usar las librerías, Maven primero buscará en el repositorio local, si no están ahí buscara en el Administrador de Repositorios de Nexus y si no irá a Maven Central y las instalará en el Nexus, cuando otra persona vuelva a compilar el código, ya no tendrá que bajarse las librerías de Maven Central, porque el Maven de otro desarrollador ya lo hizo. De esta manera tendremos una administración centralizada de nuestros artefactos producidos por el equipo de desarrollo obteniendo numerosos beneficios.

PALABRAS CLAVE:

- **REPOSITORIO DE ARTEFACTOS**
- **ADMINISTRADOR DE REPOSITORIO NEXUS**
- **APACHE MAVEN**
- **DESARROLLO DE SOFTWARE**

ABSTRACT

The DTIC, is the entity in charge of planning, designing and executing development projects, where each developer works independently with a local repository known as Maven and a remote and public repository called Maven, at the time of copying the code Maven will look in the local repository if the libraries are installed and if they are not will go to look for them in Maven Central, download and install them in our local repository, the next time Maven copy will do the same process, but as they are already downloaded will be faster and will not need to download them.

This is where the Nexus Repository Manager comes in, as FF.AA we are not always interested in depending on something external to the organization and always be able to keep control of our own libraries as they are confidential and reserved. The main idea of working with Nexus is that it is a repository of software artifacts internal to the organization and is shared by the entire development team. Having Nexus installed at the time of copying or using the libraries, Maven will first look in the local repository, if they are not there it will look in the Nexus Repository Manager and if it is not there it will go to Maven Central and install them in the Nexus, when someone else compiles the code again, you will not have to download the libraries from Maven Central, because another developer's Maven already did it. This way we will have a centralized administration of our artifacts produced by the development team obtaining numerous benefits.

KEYWORDS:

- **NEXUS REPOSITORY MANAGER**
- **REPOSITORY OF ARTIFACTS**
- **SOFTWARE DEVELOPMENT**
- **APACHE MAVEN**

CAPÍTULO I

PROYECTO TÉCNICO

1.1 Tema

Implementación de un repositorio de artefactos de software NEXUS, en un entorno de integración continua para la Dirección de Tecnologías de la Información y Comunicaciones del Ejército Ecuatoriano".

1.2 Antecedentes

La Dirección de Tecnologías de la Información y Comunicaciones del Ejército Ecuatoriano tiene la misión de planificar, diseñar y ejecutar proyectos de desarrollo y mantenimiento de software aplicativo para apoyar la gestión administrativa y operativa de las unidades militares a nivel nacional.

La DTIC está organizada en equipos de trabajo conformado por técnicos desarrolladores, analistas de tecnologías y administradores, responsables del desarrollo y mantenimiento de 11 subsistemas, 57 aplicativos y 4 servicios que forman parte del Sistema Integrado de la Fuerza Terrestre (SIFTE).

Las herramientas utilizadas para el desarrollo de software fueron establecidas en los estándares del año 2008, sin embargo, la innovación tecnológica exige que este software sea renovado periódicamente con el fin de cumplir con las mejores prácticas para la construcción de sistemas y los estándares en seguridad de la información.

En el 2017 se implementaron varias herramientas (Maven, Subversion, Fuze) para facilitar las tareas de programación y producción del software. Además, la DTIC cuenta con tres ambientes para el aseguramiento de la calidad: desarrollo, pruebas y producción.

Actualmente, se utiliza el lenguaje de programación Java, el servidor de aplicaciones Jboss y el motor de base de datos Oracle como estándares de desarrollo

de la Fuerza Terrestre; aunque existen otros lenguajes de programación (PHP), servidores de aplicaciones (Apache) y bases de datos (MySQL, PostgreSQL) que han sido integrados al SIFTE por necesidades institucionales.

1.3 Planteamiento Del Problema

La Dirección de Tecnologías de la Información y Comunicaciones del Ejército Ecuatoriano, no dispone de un entorno de desarrollo integrado que facilite el desarrollo, pruebas, integración, implantación y mantenimiento del software, por lo que se requiere implementar, entre otras herramientas, un repositorio de artefactos que esté integrado con la gestión de versiones y la integración continua.

Hasta la fecha actual, no se han actualizado los procesos y estándares de desarrollo de software en la DTIC, manteniendo metodologías antiguas que hoy en día no son las más adecuadas atender eficientemente los cambios dinámicos que requieren los usuarios. A esto se asocian los siguientes problemas:

- Los desarrolladores descargan artefactos directamente del Internet sin tener un proceso previo de verificación de malware.
- Se programan componentes con funcionalidades muy similares en lugar de utilizar los existentes en un repositorio centralizado y correctamente administrado.
- No se dispone de un repositorio centralizado para el uso de todos los equipos de desarrollo.
- Se presenta dificultad para verificar si el código fuente esta optimizado y no presenta errores en su programación.

Por lo tanto, es de vital importancia que la Dirección de Tecnologías de la Información y Comunicaciones de la Fuerza Terrestre, implemente un repositorio de

artefactos de software en un entorno de integración continua como es Nexus gestionando de una manera eficiente el proceso de fabricación de software.

1.4 Justificación

Este proyecto surge de las necesidades identificadas en el equipo de desarrolladores de la Dirección de Tecnologías de la Información y Comunicaciones del Ejército Ecuatoriano y se pretende implementar un repositorio de artefactos de software en un entorno de integración continua (NEXUS) para mejorar la calidad y tiempos en las actividades y mantenimiento de software.

La integración continua (IC) es una buena práctica que permite a los desarrolladores automatizar y agilizar determinados procesos como así también eliminar los inconvenientes causados a menudo por largos ciclos de integración: construcciones de código que fallan, integración manual de código y largas pruebas de regresión entre otros.

Al final del proyecto se busca tener la administración centralizada de artefactos, mediante la implementación de un repositorio compartido de artefactos que faciliten las tareas de los equipos de desarrollo de la DTIC.

Los beneficiarios directos son los equipos de desarrollo de la DTIC e indirectamente se beneficiarán a 25.000 usuarios de los aplicativos, sistemas y servicios de la Fuerza Terrestre. Se debe mencionar los beneficios que tendrán la implementación de Nexus para el desarrollo del software.

1.5 Objetivos

1.5.1 Objetivo General

- Implementar un repositorio de artefactos de software en un entorno de integración continua (NEXUS) para la Dirección de Tecnologías de la Información y Comunicaciones de la Fuerza Terrestre.

1.5.2 Objetivos Específicos

- Planificar la migración de las herramientas que actualmente dispone la DTIC como repositorio de artefactos a NEXUS.
- Respaldar el repositorio de artefactos.
- Instalar, implementar y configurar las herramientas Maven y Nexus en el entorno de Pruebas de la DTIC.

1.6 Alcance

El presente trabajo investigativo tiene como alcance es estudio, análisis, implementación y configuración de un Administrador de Repositorios de Nexus en el entorno de pruebas con el fin de mejorar el trabajo del equipo de desarrolladores de la Dirección de Tecnologías de la Información y Comunicaciones de la Fuerza Terrestre.

Adicional servirá de fuente de información y consulta para todas aquellas personas relacionadas o interesadas en el tema.

CAPÍTULO II

MARCO TEÓRICO

2.1 ¿Qué es la Ingeniería de Software?

Un sistema de software, denominado también aplicación o simplemente software, es un conjunto integrado de programas que en su forma definitiva se pueden ejecutar, pero comprende también las definiciones de estructuras de datos que utilizan estos programas y también la documentación referente a todo ello tanto la documentación de ayuda en el uso del software para sus usuarios como la documentación generada durante su construcción, parte de la cual también servirá para su mantenimiento posterior. (Campderrich, 2013)**.**

2.2 ¿Qué es Software?

La palabra software es un vocablo ingles haciendo referencia aciertos aplicativo de informática, es decir es la parte lógica de una computadora

El software está compuesto por los programas o conjunto de programas diseñados para cumplir ciertas funciones dentro del sistema de un computador

Software es mucho más que un código de programa. Un programa es un código ejecutable, usado para propósitos computacionales. El Software se considera una colección de códigos ejecutables de programación, asociada a las bibliotecas y a la documentación. El Software, cuando se ha hecho para cubrir requisitos específicos se llama producto software.

2.3 Importancia del Software

El software es una de las áreas que avanzado abismalmente con el avance de la tecnología y es la clave del éxito para muchas empresas y negocios. El software es imprescindible para cualquier sistema informático, puesto que, sin él, este no funcionaría.

El desarrollo de software es la industria con mayor crecimiento vertical en los últimos años ya que casi en todas las actividades cotidianas está inmerso el desarrollo de software, ya que permite construir sistemas complejos, inteligencia, artificial automatización de maquinarias, facilitando de esta manera el trabajo diario a la humanidad y a veces llegando a remplazar al ser humano por robot.

2.4 Clasificación del Software

El software pude clasificarse en tres grandes grupos:

2.4.1 Software de Sistemas

Se llama software de sistemas o software de base, al conjunto de programas escritos necesarios para dar servicio a otros programas e interactuar con el sistema operativo. El área de software de sistemas se caracteriza por: gran interacción y control sobre el hardware de la computadora, uso intensivo por parte de usuarios múltiples, operación concurrente que requiere la secuenciación, recursos compartidos y administración de un proceso sofisticado, estructuras complejas de datos e interfaces externas múltiples. (Pressman, 2010).

Entre los más conocidos Software de sistemas tenemos los sistemas operativos como son Windows, Mac OS y Linux, herramientas de diagnóstico, controladores de dispositivos, entorno de escritorio, BIOS, servidores de información y herramientas de Corrección y Optimización, cargadores de programas, etc.

2.4.2 Software de Programación

Es el conjunto de herramientas que permiten al programador desarrollar programas informáticos en donde se escribe el código para desarrollar un sin número de sistemas dentro de un sistema operativo, usando diferentes alternativas y lenguajes de programación, de una manera práctica. Estos incluyen básicamente:

- Editores de texto

- Compiladores
- Intérpretes
- Enlazadores
- Depuradores
- Entornos de Desarrollo Integrados (IDE).

2.4.3 Software de Aplicación

Son aquellos programas diseñados que resuelven una necesidad específica de negocios. Las aplicaciones en esta área procesan datos comerciales o técnicos en una forma que facilita las operaciones de negocios o la toma de decisiones administrativas o técnicas. Entre las que más se destacan tenemos base de datos, software empresarial, software de diseño asistido, aplicaciones ofimáticas, software de control y calculo, etc. (Pressman, 2010).

2.5 Problemas de Software

2.5.1 La Calidad y Productividad

A pesar d ellos grandes adelantos la calidad y productividad de su proceso de elaboración todavía no alcanzado su nivel máximo, en cuanto a la calidad la causa principal de las dificultades es la complejidad del software comparado con otros productos, por otra parte, la productividad el factor que tiene peso realmente importante es la baja productividad, a diferencia de otras tecnologías, en un proyecto de software de desarrollo empieza desde cero. (Campderrich, 2013).

2.5.2 El Aprovechamiento de Elementos en la Fabricación en Serie

Uno de los grandes retos por el momento de los desarrolladores de software sea conseguir desarrollar componentes de software que sean reutilizables y reutilizar estos componentes ya que estarían probados y mejorar la calidad del software producido.

2.6 Características Básicas del Software

El software es elemento de un sistema lógico y no de uno físico. Por tanto, tiene características que difieren considerablemente de las del hardware. (Pressman, 2010).

- **El software se desarrolla o modifica con intelecto; no se manufactura en el sentido clásico. -** Aunque hay algunas similitudes entre el desarrollo de software y la fabricación de hardware. En ambas, la alta calidad se logra a través de un buen diseño, pero la fase de manufactura del hardware introduce problemas de calidad que se corrigen con facilidad en el software. Las dos actividades requieren la construcción de un "producto", pero los enfoques son distintos. Los costos del software se concentran en la ingeniería. Esto significa que los proyectos de software no pueden administrarse como si fueran proyectos de manufactura. (Pressman, 2010).
- **El software no se "desgasta".** El hardware presenta una tasa de fallas relativamente elevada en una etapa temprana de su vida son atribuibles a defectos de diseño, los defectos se corrigen y la tasa de fallas se abate a un nivel estable durante cierto tiempo. No obstante, conforme pasa el tiempo, la tasa de fallas aumenta de nuevo a medida que los componentes del hardware resienten los efectos acumulativos de suciedad, vibración, abuso, temperaturas extremas y muchos otros inconvenientes ambientales. En pocas palabras, el hardware comienza a desgastarse. El software no es susceptible a los problemas ambientales que hacen que el hardware se desgaste. (Pressman, 2010).
- **Aunque la industria se mueve hacia la construcción basada en componentes. -** La mayor parte del software se construye para un uso individualizado. (Pressman, 2010).
- **Permite interactuar con múltiples sistemas**. - El software debe

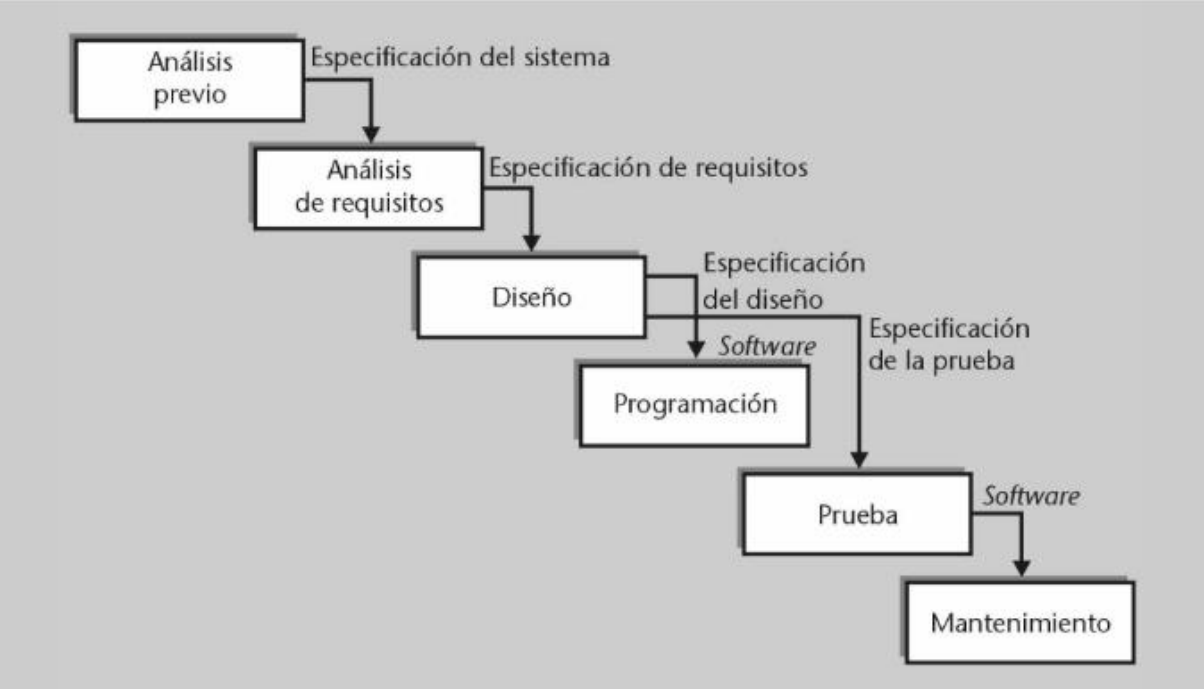
ser compatible**,** es decir, fácil para ejecutar las tareas necesarias para optimizar y acelerar el rendimiento del equipo e integrar los diferentes sistemas.

- **Diversos lenguajes de programación. -** Los diferentes lenguajes de programación funcionan como intermediarios entre el hardware y el ser humano simplificando las distintas instrucciones a través de un lenguaje lógico - técnico e interpretado por la maquina en forma binaria.

2.7 Ciclo de Vida del Software (SDLC)

Figura 1:

Ciclo de vida del software



Nota: La figura muestra el ciclo de vida de software. Figura tomada de tutorialsPoint, 2019).

Según (TutotialsPoint , 2019) define al Software Development Life Cycle (SDLC) traducido al español como ciclo de vida del software, es un proceso utilizado por la industria del software para diseñar, desarrollar y probar softwares de alta calidad. El

SDLC tiene como objetivo producir un software de alta calidad que cumpla o supere las expectativas del cliente, llegue a su finalización en tiempos y estimaciones de costos,

Existen estándares internacionales para el ciclo de vida de software en este caso es la norma ISO / IEC 12207 cuyo objetivo es ser el estándar que define las tareas necesarias para desarrollar y mantener software.

2.7.1 ¿Qué es el SDLC?

Según (TutotialsPoint , 2019), SDLC es el proceso que se lleva a cabo para un proyecto de software seguido por los equipos de desarrollo de software, consiste en describir cómo se va a desarrollar, mantener, reemplazar o mejorar un software,

De igual se define la metodología a seguir para mejorar la calidad de software y el proceso de desarrollo general, en cada etapa se obtienen documentos que son las bases de partida de la etapa siguiente es decir no se puede comenzar la siguiente etapa sin haber terminado la anterior.

2.7.2 Etapas del Ciclo de Vida del Software

a. Planificación y análisis del sistema. – En esta etapa se definen los grandes rasgos del nuevo sistema e intentan crear su mejor modelo de software, las limitaciones del producto, adaptación a un entorno de hardware y red determinado, la forma de intercambiar información con otros sistemas. Además, es necesario considerar los recursos necesarios para el desarrollo del software y condicionamientos temporales, tiempo disponible para entregar el producto terminado y evaluar la viabilidad técnica, económica y legal del proyecto a desarrollar.

El documento resultante de esta etapa se denomina especificaciones del sistema y sirve como base para tomar la decisión definitiva sobre la continuidad del proyecto. (Campderrich, 2013).

b. Definición y análisis de requisitos. Su objetivo es definir con detalle las necesidades de información que tendrá que resolver el software sin tener en cuenta por el momento los medios técnicos con los que se tendrá que llevar a término el desarrollo de software, como es el lenguaje de programación, el gestor de base de datos, los componentes que se puede reutilizar. En esta etapa pensamos en el software que es necesario desarrollar y sus interfaces con el entorno. El desarrollador debe tener conocimientos generales sobre el dominio de la aplicación y obtener información de los usuarios y otras fuentes que le permitan hacer una idea precisa de las funciones y requisitos en general. (Campderrich, 2013).

Esto se realiza mediante las SRS (especificación de requisito de software) donde consta todos los requisitos del producto que se diseñan y desarrollan durante el ciclo de vida del proyecto.

c. Diseño y arquitecta del producto. si el análisis especifica el problema el diseño especifica una solución a este problema. Hay que diseñar varios aspectos diferenciados como su arquitectura general, las estructuras de datos, la especificación de cada programa y las interfaces con el usuario, que se tiene que llevar a cabo de manera que se pueda codificar el software y se documenta en las especificaciones de documento de diseño.

Es de vital importancia revisar las especificaciones de documento d diseño en función a parámetros como le evaluación de riesgos, el modularidad del diseño, la solidez del producto, el presupuesto y limitaciones de tiempo para la entrega.

El documento resultante de la especificación del diseño es el mejor momento para elaborar las especificaciones de pruebas que describe que datos tiene que probar cada programa o grupo de programas y cuáles son los resultados esperados en cada caso. (Campderrich, 2013).

d. Codificación. Consiste en traducir el diseño o código y comienza la construcción y el desarrollo real del producto de software.

Los desarrolladores deben seguir pautas de codificación definidas por la organización empleando las herramientas de programación como copiladores, intérpretes y depuradores, el entregable que se genera en esta etapa es el programa con todas sus funcionalidades y componentes. (Campderrich, 2013).

e. Pruebas del producto. Esta etapa es un subconjunto de todas las etapas, consiste en probar el software desde distintos puntos de vista de una manera planificada y natural, localizar y corregir errores dentro del software hasta que el producto alcance los estándares de calidad definidos en las SRS. (Campderrich, 2013).

f. Despliegue en el marcado y mantenimiento. Pasadas las pruebas del producto está listo para ser implementarse, se puede lanzar en un segmento limitado, luego en función de los comentarios puede lanzarse con mejoras sugeridas. Su mantenimiento se lo realiza en base de sus clientes existentes.

2.8 Reutilización de Software

2.8.1 Definición

La reutilización de software aparece como una alternativa para desarrollar aplicaciones y sistemas de software de un manera más eficiente, productiva y rápida, en base a softwares ya desarrollados anteriormente, la idea es reutilizar elementos y componentes de software en lugar de tener que desarrollarlos desde un principio(Unknown, 2012).

Entre las principales ventajas de reutilizar software destacan como el de reducir tiempo de desarrollo, reducir costos, incrementar productividad, mayor fiabilidad, mayor eficiencia y consistencia, además existen dificultades al momento de reutilizar software debido a que muchas empresas no lo consideran importante ya que se debe invertir

antes de ver los resultados y no existe información de las ventajas existentes al reutilizar software.

2.8.2 Beneficios y Costes de la Reutilización de Software

Existen dos enfoques en cuanto a la reutilización de software el enfoque oportunista que aprovecha los elementos del software construidos en proyectos anteriores, pero no fueron construidos con ese objetivo y un enfoque más planificado o proactivo que se construyeron con el objetivo de ser reutilizados lo que puede significar en el momento de desarrollo mayor inversión de recursos.

2.9 Artefacto de Software

2.9.1 Definición

Un artefacto de software es un producto tangible, que está ligado a los métodos o procesos de desarrollo específicos de software, también se puede definir que son el resultado de trabajo parcial o final que es producido y usado durante un proyecto. Los artefactos son usados para capturar y llevar la información del proyecto.

Artefactos tales como diagrama de clases, casos de uso, entre otros modelos UML colaboran en la descripción de la función, la arquitectura o diseño de software, mientras que otro grupo de artefactos como es el caso de planes de proyecto, casos de uso o enfoque de riesgos se enfocan en el proceso de desarrollo de software, el código fuente también se considera como artefacto de software, en vista que los ejecutables es necesario para el plan de testeo.

Los artefactos detallan el diseño pretendido al inicio del proyecto mientras que otros artefactos se conservan a lo largo del ciclo de vida con información y nuevas características que se van añadiendo durante el desarrollo del sistema.

- Según el blog, (Proceso de Desarrollo de Software, 2012). Un artefacto puede ser:

- **Un documento**: como un Caso de Negocio o un documento de la arquitectura del Software.
- **Un modelo**: como un modelo de caso de uso.
- **Un elemento** de un modelo: como una sola clase de todo el Diagrama de Clases.

Los modelos y elementos de modelos, tienen reportes asociados a ellos. Un reporte saca información acerca de los modelos y sus elementos mediante una herramienta. Un reporte presenta un artefacto o un conjunto de artefactos. La mayoría de los artefactos tienen directrices, las cuales describen los artefactos con más detalle. (Proceso de Desarrollo de Software, 2012).

Para hacer el desarrollo de un sistema de Software manejable completo, los artefactos están organizados en conjuntos correspondientes a las disciplinas. Muchos artefactos son usados en varias disciplinas; por ejemplo, la Lista de Riesgos, el Documento de Arquitectura del Software y el Plan de iteración. Este tipo de artefactos pertenecen al conjunto de artefactos donde ellos fueron originalmente creados. (Proceso de Desarrollo de Software, 2012).

2.10 Metodologías de Software para la Construcción de Artefactos

2.10.1¿Qué es una Metodología?

Una metodología de desarrollo de software se define como un conjunto de procedimientos o instrucciones para estructurar, planificar y controlar el proceso de desarrollo de software, teniendo en cuenta factores como los costes, la calidad, la planificación y las dificultades asociadas al desarrollo de software, es decir se trata del proceso que se debería seguir al momento de diseñar un programa de software específico.

Una metodología se centra en la comunicación, manipulación e intercambio de información y datos entre las personas comprometidas en el desarrollo con el objetivo de desarrollar programas siguiendo una metodología, reglas, guías, criticas y sugerencias de diseño.

Una metodología define una estrategia global para enfrentarse con el proyecto. Entre los elementos que forman parte de una metodología se pueden destacar:

- Fases: tareas a realizar en cada fase o etapa.
- Productos: Entrada y salida de cada fase, documentos.
- Procedimientos y herramientas: apoyo a la realización de cada tarea.
- Criterios de evaluación: del proceso y del producto. Saber si se han logrado los objetivos. (Maida & Pacienzia, 2015).

Existen tres tipos de metodologías las conocidas como tradicionales, las metodologías agiles y las metodologías hibridas que es la combinación de las dos metodologías descritas anteriormente.

2.10.2 Metodología de Desarrollo Tradicional

Son conocidas también como metodologías pesadas ya que se centran en documentar todo el proyecto, la planificación y el control del mismo, en especificaciones precisas de requisitos y en cumplir con un plan de trabajo, definido todo esto, en la fase inicial del desarrollo del proyecto. Estas metodologías imponen una disciplina rigurosa del trabajo sobre el proceso de desarrollo del software, definiendo roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada, otra característica importante son los altos costos en implementar un cambio y la poca flexibilidad en proyectos donde sus requerimientos pueden cambiar constantemente. (Maida & Pacienzia, 2015).

2.10.3 Tipos de Metodologías Tradicionales

a. Waterfall (Cascada). – Se denomina así por la posición de sus fases en el desarrollo de esta, que parecen caer en cascada hacia las siguientes fases. Es el enfoque metodología que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase. Una desventaja es que los resultados no se pueden ver hasta muy avanzado el proyecto por lo que cualquier cambio debido a un error puede suponer un gran retraso además de un alto coste de desarrollo. (Maida & Pacienzia, 2015).

b. Incremental. - Permite construir el proyecto en etapas incrementales en donde cada etapa agrega funcionalidad. Estas etapas, consisten en requerimientos, diseño, codificación, pruebas y entrega. Permite entregar al cliente un producto más rápido en comparación del modelo en cascada. (Maida & Pacienzia, 2015)

Esta metodología reduce los riegos ya que provee visibilidad sobre el progreso de las nuevas versiones y los clientes no deben esperar hasta el fin del desarrollo para poder utilizar el sistema, ya que pueden utilizarlos desde el primer incremento, las pruebas y la integración son constantes, la solución se va mejorando en forma progresiva a través de las múltiples iteraciones. Una de las principales desventajas es que requiere de mucha planeación, tanto administrativa como técnica, metas claras para conocer el estado del proyecto. (Gabriel, s. f.).

c. Las metodologías de desarrollo ágiles: Son aquellos métodos de desarrollo interactivo e incremental de acuerdo a la necesidad y dificultad del producto que se quiere construir, estas metodologías permiten adaptar la forma de trabajo a las

condiciones del proyecto, en colaboración con equipos multidisciplinarios y organizados con el objetivo de conseguir flexibilidad e inmediatez a la hora de construir software. Esta metodología ha demostrado ser efectiva en proyectos con requisitos muy cambiantes, y están revolucionando la forma de producir software.

2.10.4 El Manifiesto Ágil

Según (Canos, Hetelier, & Penadés, 2006) El manifiesto ágil comienza enumerando los principales valores del desarrollo ágil. Se valora:

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. – el personal involucrado es el principal factor de éxito de un proyecto de software, e éxito no está asegurado, sin embargo, si el equipo de trabajo funciona es más fácil conseguir el objetivo final.
- Desarrollar software que funciona más que conseguir una buena documentación. – No producir documentación a menos que sean necesarios de forma inmediata para tomar una decisión importante, estos documentos deben ser cortos y centrarse en lo fundamental.
- La colaboración con el cliente más que la negociación de un contrato. – La interacción constante con el cliente y el equipo de desarrollo será la pauta que marque la marcha del proyecto y asegure el éxito del mismo.
- Responder a los cambios más que seguir un plan estrictamente. – La posibilidad de adaptarse a los cambios que se puedan presentar durante el desarrollo del proyecto determina el éxito o fracaso del mismo.

2.10.5 Metodologías Ágiles

a. SCRUM: creada por Ken Schwaber, define un marco para la gestión de proyectos con un rápido cambio de requisitos, su finalidad es la entrega de valor en cortos periodos de tiempo y se basa en tres pilares que son la trasparencia, inspección

y adaptación. Esta metodología se enmarca en la flexibilidad para integrar y adaptar a nuevos requisitos durante el desarrollo del proyecto, en el factor humano y colaboración e interacción con el cliente para poder obtener buenos resultados.

b. Cristal metodologías: Esta metodología ágil fue creada por Cockburn, se centra en las personas y sus interacciones cuando se desarrolla un proyecto en lugar de centrarse en sus procesos y herramientas, la forma de comunicación entra las personas involucradas en el desarrollo del proyecto tienen un mayor impacto en el resultado del proyecto.

Esta metodología se centra en que los equipos pueden optimizar sus procesos como su trabajo y convertirse en un equipo optimizado y los proyectos son únicos y dinámicos y requieren métodos específicos.

c. Programación extrema XP. – La a metodología XP o programación extrema es una metodología ágil y flexible utilizada para la gestión de proyectos y desarrollo de software, centrada en potenciar las relaciones interpersonales y la retroalimentación continua entre el cliente y el equipo de desarrollo, es muy sencilla y fácil de implementar, se considera que el software funciona por encima de una buena documentación.

2.10.6 Metodologías Híbridas

Este tipo de metodología está marcando la tendencia en el área de Ingeniería de Software ya que combina las mejores prácticas de las metodologías tradicionales y metodologías agiles, generando un incremento en la productividad, entre las más conocidas y utilizadas tenemos:

a. Metodología EssUP (Essential Unnified Process). – Consiste en la mescla de la metodología RUP(Rationall Unnified Process) con la metodología ágil Scrum considerando las mejores características d las dos metodologías para mejorar el

proceso de desarrollo de software, involucra tres campos de procesos principales, el campo de proceso unificado, en campo de los métodos agiles y el campo de la madures del proceso, aportando diferentes capacidades como es estructura, agilidad y mejora los procesos de desarrollo. (Andrango, 2017)

b. Metodología Híbrida SCRUM/XP. - Resulta de la mezcla de las metodologías Scrum con XP, es una opción factible para el desarrollo de software, ya que Scrum es una de las metodologías ágiles más utilizadas por grandes empresas de construcción de software a nivel mundial, mientras que la programación extrema XP es una metodología ligera para el desarrollo de software que propicia un conjunto de valores y prácticas para aumentar la productividad, manteniendo una buena comunicación o una retroalimentación entre el cliente y el equipo de desarrollo enfrentando cambios que surgen durante el transcurso del desarrollo del proyecto. (Andrango, 2017).

2.11 Normas ISO 25000

La calidad del producto, junto con la calidad del proceso, es uno de los aspectos más importantes actualmente en el desarrollo de Software. Las normas ISO/IEC 25000 constituye una serie de normas basadas en ISO/IEC 9126 y en ISO/IEC 14598 cuyo objetivo principal es guiar el desarrollo de los productos de software mediante la especificación de requisitos y evaluación de características de calidad. (ISO 25000, 2019)

2.11.1 Norma ISO/IEC 25010

El modelo de calidad o norma ISO/IEC 25010, establece el sistema para la evaluación de la calidad del producto. En este modelo se determinan las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto software determinado. (ISO 25000, 2019).

La calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. Son precisamente estos requisitos tales como funcionalidad, rendimiento, seguridad, mantenibilidad, etc. Los que se encuentran representados en el modelo de calidad, el cual categoriza la calidad del producto en características y subcaracterísticas. (ISO 25000, 2019).

Figura 2:

Calidad del producto software.

Nota: El modelo de calidad del producto definido por la ISO/IEC 25010 compuesto por ocho características de calidad. Figura tomada de (ISO 25000, 2019).

A continuación, voy a detallar cada una de las características:

a. Adecuación Funcional. – Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades del cliente.

b. Eficiencia del desempeño. - Se refiere a la cantidad de recursos utilizados bajo determinadas condiciones para medir el desempeño, la capacidad de producto y el comportamiento temporal del mismo.

c. Compatibilidad. – Es la capacidad de más de dos sistemas intercambiar información entre sí y cumplir con ciertas funciones cuando comparten el mismo entorno de hardware y software.

d. Usabilidad. – Capacidad del producto de software para poder ser entendido, aprendido, usado y resultar atractivo para el usuario cuando se usa bajo determinadas condicione. (ISO 25000, 2019).

e. Fiabilidad. Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados. (ISO 25000, 2019).

f. Seguridad. - Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos. (ISO 25000, 2019).

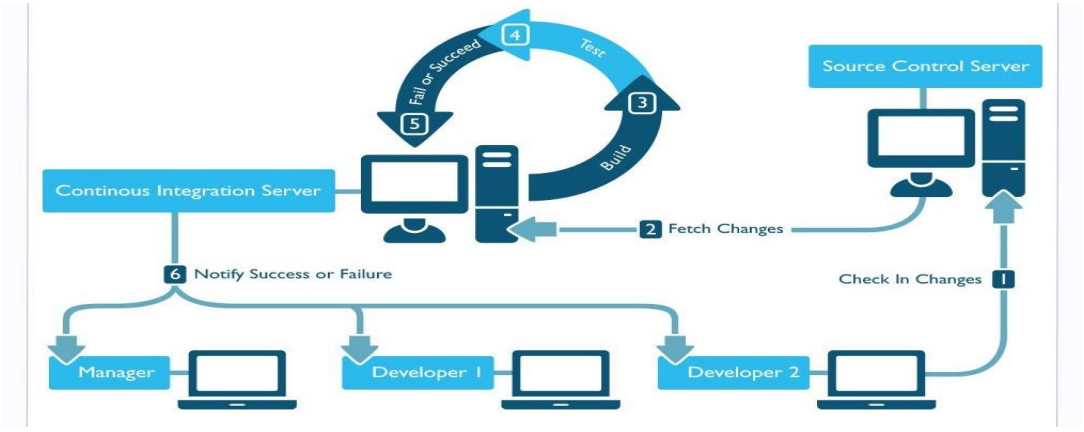
g. Mantenibilidad. - Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas. (ISO 25000, 2019).

h. Portabilidad. - Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro. (ISO 25000, 2019).

2.12 Integración Continua

Figura 3:

Proceso de integración continua



Nota: La figura muestra el proceso que sigue la integración continua. Figura tomada de (Federico Toledo 2017).

2.12.1 ¿Qué es la Integración Continua?

La integración continua es una práctica de desarrollo de software donde los miembros de un equipo integran su trabajo con frecuencia, generalmente cada persona se integra al menos diariamente, lo que lleva a múltiples integraciones por día. Cada integración se verifica mediante una compilación automatizada para detectar errores de integración lo más rápido posible. Muchos equipos encuentran que este enfoque conduce a problemas de integración significativamente reducidos y permite que un equipo desarrolle software cohesivo más rápidamente. (Flower, 2006).

La integración continua (CI) es una fase en el ciclo de desarrollo de software donde el código de diferentes miembros del equipo o diferentes características se integran entre sí. Esto generalmente implica fusionar código (integración), construir la aplicación y realizar pruebas básicas, todo dentro de un entorno efímero. (Medium Corporation , 2020).

En el pasado, el código se integraba en una “fase de integración” del ciclo de vida del desarrollo de software, los equipos de desarrollo trabajaban por separado dedicados a construir diferentes aplicaciones o servicios lo que suponía un dolor de cabeza al momento de integrar, encontrar los posibles errores lo que ocasionaba perdida de recurso económicos, de personal y tiempo. (Medium Corporation , 2020).

Según (Galvan, s.f.) menciona que al implementar esta práctica exitosamente involucra ciertos requisitos como:

- Tener un repositorio maestro donde esté disponible todo el código fuente y del que cualquier integrante del equipo pueda obtenerlo.
- Automatizar el proceso de integración para que cualquier persona pueda

generar una versión ejecutable del sistema a partir del código fuente.

- Automatizar las pruebas para que sea posible ejecutar la matriz de pruebas en cualquier momento con un solo comando.
- Asegurar que cualquiera puede obtener el ejecutable más reciente y tener la confianza de que es la mejor versión hasta el momento.

Todo esto requiere bastante disciplina y requiere un esfuerzo significativo para introducirlo a un proyecto, pero una vez habilitado es sencillo mantenerlo y brinda grandes beneficios.

2.12.2 Beneficios de la Integración Continua

- Reducción de riesgo.
- Detección temprana de errores y fallas.
- Fomenta el trabajo minucioso y la comunicación constante.
- Disponibilidad de una versión del código funcional y actualizada.
- Aumento de la moral del equipo de desarrollo.
- Reducción de costos.

2.12.3 Inconvenientes de la Integración Continua

- Degeneración de la arquitectura.
- Cambio de los procesos habituales.
- Necesidad de un servidor.
- Elaboración de plan de pruebas propio.
- No todos utilizan los repositorios correctamente.
- Temor a dañar el código de la línea principal.

Según (Medium Corporation , 2020), los flujos de trabajo de CI varían mucho según las herramientas, el lenguaje de programación, el proyecto y muchos otros factores, pero un flujo común tiene estos pasos.

- Empujar al repositorio de código. - Normalmente, habría un repositorio de código y algún tipo de flujo de trabajo para contribuir con el nuevo código.
- Análisis estático. - se realiza en la base de código de la aplicación sin la necesidad de ejecutar el software.
- Pruebas previas a la implementación. - Debe realizarse cualquier prueba que se pueda ejecutar sin implementar en el servidor. Sean pruebas unitarias, funcionales o de integración.
- Empaquetado e implementación en el entorno de prueba. - Dependiendo del tipo de proyecto, la aplicación se crea, empaqueta, envía a un entorno de prueba o preparación.
- Pruebas posteriores a la implementación. - Necesitan que se implemente la aplicación, en esta fase es donde se ejecutan y varían según las herramientas.

2.12.4 ¿Qué es la Entrega Continua y por qué es Útil?

La entrega continua es una extensión de la integración continua, enfocada en automatizar el proceso de entrega de software para que lo equipos puedan implementar fácilmente.

2.13 Conceptos Clave y Prácticas para Procesos Continuos

Existen conceptos fundamentales que debemos tener en cuenta para llevar a cabo un proceso de integración, la entrega y la implementación continua.

Pequeños cambios interactivos.

2.13.1 Definición de Metadatos

Este término apareció en los años 60 con el objetivo de describir un conjunto de datos, pero en la actualidad está siendo muy utilizado en el mundo virtual. La etimología de este término consta de dos palabras, una griega y otra latina. Por un lado, la palabra griega “meta”, que significa después de o más allá de, y por otro lado el vocablo latino

“datum”, que significa dato. Así mismo, se forma la expresión metadatos como más allá de los datos. En base a esto, metadatos son un conjunto de datos que describen el contenido informativo de un recurso, de archivos o de información de los mismos. Es decir, es información que describe otros datos. No obstante, la palabra metadatos no tiene una definición exclusiva, ya que son varias las expresiones con las que se conoce, tales como informaciones sobre datos, informaciones sobre informaciones o datos sobre informaciones. (PowerData, 2008).

El término metadato se emplea en el amito de la informática, el interés ha suscitado en las empresas que requieren o disponen de grandes cantidades de información para ello los metadatos ayudan a gestionar los datos e información para aumentar el rendimiento y efectividad de la empresa.

Según (PowerData, 2008), los metadatos poseen una característica principal que son multifuncionales, facilitando una buena administración de los datos, entre las principales funciones de los metadatos tenemos las siguientes:

- Facilitan la búsqueda y el análisis de los datos desde la propia fuente,
- Mejora la gobernanza de loso datos en un entorno estandarizado
- Ayuda a la integración completa mejorando la gestión de datos de manera global
- Facilita la estandarización y que elimina errores y debilidades, mejorando la calidad de los metadatos durante el ciclo de vida.
- Mejora los informes ayudando a gestionar los cambios y brindando una mayor seguridad en los datos críticos de una empresa.

2.13.2 El Ciclo de Vida de los Metadatos

Los metadatos tienen un ciclo de vida que va detallando paso a paso cada etapa por la que pasa, se puede diferenciar tres etapas diferentes que son creación, manipulación y destrucción.

a. Creación. - Es donde se crean los metadatos en forma manual, automática y semiautomática.

b. Manipulación. – Es donde se llevan los cambios en diferentes puntos,

c. Destrucción. - En este punto es donde se procede a la destrucción de los metadatos.

2.13.3 ¿Qué es una Librería de Software?

Según la definición de (GARZAS, 2014), una librería como a un conjunto reutilizable de recursos, funciones, métodos, código ya implementado, que podemos agrupar en un solo fichero.

Maven es una herramienta muy poderosa que nos permite a más de copilar código también nos permite la gestión de librerías, ya que al momento de programar los desarrolladores realiza el código con la finalidad de ahorrar tiempo y nos faciliten el desarrollo.

2.13.4 Componentes

Según (Sonatype, 2008)**,** En Nexus Repository Manager, un componente es un recurso como una biblioteca, un artefacto, paquete, archivo, binario, contenedor, un marco y otros términos, utilizados por su aplicación de software en tiempo de ejecución, integración o tiempo de ejecución de pruebas unitarias, o requerido como parte de su proceso de compilación. Podría ser una aplicación completa o un recurso estático como una imagen. Por lo general, estos componentes son archivos de una gran variedad de archivos que incluyen.

Los componentes proporcionan todos los componentes básicos y características que permiten a su equipo de desarrollo crear aplicaciones. Se ha vuelto común crear aplicaciones combinando las características de los componentes de código abierto con sus propios componentes personalizados para crear una aplicación para un dominio

específico. Uno de los muchos beneficios del desarrollo de software basado en componentes y de código abierto es la reducción del retrabajo.

Los componentes pueden estar compuestos por múltiples componentes anidados y proporcionan todos los componentes básicos y las características que permiten que un equipo de desarrollo implementar aplicaciones robustas ensamblándolas y agregando sus propios componentes relacionados con el negocio para crear una aplicación potente y completa. Los componentes también pueden ser el ancla para otros metadatos. (Sonatype, 2008).

2.13.5 Servidor de Aplicaciones

Un servidor de aplicaciones, es un software que proporciona un entorno para ejecutar aplicaciones web. La mayoría de los servidores de aplicaciones también proporcionan funcionalidad a las aplicaciones web que se ejecutan en su entorno a través de un conjunto de API. Por ejemplo, un servidor de aplicaciones puede proporcionar una API para conectarse a una base de datos. (Red Hat Customer Portal , 2020).

2.13.6 Java Enterprise Edition 8

Java Platform, Enterprise Edition (Java EE), es una plataforma empresarial basada en estándares que proporciona un entorno de API y tiempo de ejecución para ejecutar y desarrollar aplicaciones Java. El objetivo es mejorar la productividad del desarrollador al proporcionar capacidades empresariales ricas en marcos fáciles de consumir que eliminen las repeticiones y reduzcan la carga técnica. Los marcos que componen Java EE se prueban en combinación. (Red Hat Customer Portal , 2020).

2.14 JBoss EAP 7

Red Hat JBoss Enterprise Aplicación Platform (JBoss EAP) 7.3 es una implementación certificada de la especificación Java Enterprise Edition (Java EE) 8. Las

versiones principales de JBoss EAP se bifurcan del proyecto comunitario WildFly en ciertos puntos cuando el proyecto comunitario ha alcanzado el nivel de integridad de características deseado. Después de ese punto, se lleva a cabo un período prolongado de pruebas y producción en el que JBoss EAP se estabiliza, certifica y mejora para su uso en producción. Durante la vida útil de una versión principal de JBoss EAP, las características seleccionadas pueden ser seleccionadas y trasladadas desde el proyecto comunitario a una serie de lanzamientos menores que mejoran las características dentro de la misma familia de versiones principales. (Red Hat Customer Portal , 2020).

JBoss EAP proporciona opciones preconfiguradas para características tales como clustering de alta disponibilidad, mensajería y almacenamiento en caché distribuido. También permite a los usuarios escribir, implementar y ejecutar aplicaciones utilizando las diversas API y servicios que proporciona JBoss EAP. (Red Hat Customer Portal , 2020).

JBoss EAP incluye una estructura modular que permite habilitar el servicio solo cuando es necesario, mejorando la velocidad de inicio. La consola de administración basada en la web y la interfaz de línea de comando de administración (CLI) hacen innecesaria la edición de archivos de configuración XML y agregan la capacidad de crear scripts y automatizar tareas. Además, JBoss EAP incluye API y marcos de desarrollo para desarrollar rápidamente aplicaciones Java EE seguras y escalables. (Red Hat Customer Portal , 2020).

2.14.1 Características de JBoss EAP

Java EE. - JBoss EAP es compatible con Java Enterprise Edition 8 plataforma completa y perfil web certificado.

a. Dominio administrado. – Administración centralizada de múltiples instancias de servidor y hosts físicos, mientras que un servidor independiente permite una sola instancia de servidor. Gestión centralizada y simplificada de la seguridad de las aplicaciones.

b. Consola de administración y CLI de administración. - Nuevo dominio o interfaces de administración de servidor independientes. La CLI de administración puede automatizar las tareas de administración.

c. Diseño de directorio simplificado. - El directorio de módulos contiene todos los módulos del servidor de aplicaciones. (Red Hat Customer Portal , 2020)

d. Mecanismo de carga de clase modular. - Los módulos se cargan y descargan a pedido. Esto mejora el rendimiento, tiene beneficios de seguridad y reduce los tiempos de arranque y reinicio.

e. Gestión simplificada de la fuente de datos. – Las fuentes de datos se crean y administran utilizando la consola de administración y la CLI de administración.

Marco de seguridad unificado. - Elytron proporciona un marco unificado único que puede administrar y configurar el acceso tanto para servidores independientes como para dominios administrados.

2.14.2 Subsistemas

Muchas de las API y capacidades que están expuestas a las aplicaciones implementadas en JBoss EAP están organizadas en subsistemas. Los administradores pueden configurar estos subsistemas para proporcionar un comportamiento diferente, según el objetivo de la aplicación. Por ejemplo, si una aplicación requiere una base de datos, se puede configurar un origen de datos en el datasources subsistema y acceder a esa aplicación después de que se implemente en ese servidor o dominio JBoss EAP. (Red Hat Customer Portal , 2020).

a. Alta disponibilidad. - La alta disponibilidad en JBoss EAP se refiere a varias instancias de JBoss EAP que trabajan juntas para proporcionar aplicaciones que son más resistentes a las fluctuaciones en el tráfico, la carga del servidor y la falla del servidor. Alta disponibilidad incorpora conceptos como escalabilidad, equilibrio de carga y tolerancia a fallas. (Red Hat Customer Portal , 2020).

b. Modos de funcionamiento. - Además de proporcionar funcionalidad y API a sus aplicaciones, JBoss EAP tiene potentes capacidades de administración que difieren según el modo operativo que se use para iniciar JBoss EAP. También ofrece un modo operativo de servidor independiente para administrar instancias discretas y un modo operativo de dominio administrado para administrar grupos de instancias desde un único punto de control. (Red Hat Customer Portal , 2020).

2.15 ¿Qué es GIT?

2.15.1 ¿Qué es el Control de Versiones?

Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante, permite revertir los archivos seleccionados a un estado anterior, revertir todo el proyecto a un estado anterior, comparar los cambios a lo largo del tiempo, ver quién modificó por última vez algo que podría estar causando un problema, quién presentó un problema y cuándo, y más. Usar un VCS también generalmente significa que, si arruinas las cosas o pierdes archivos, puedes recuperarte fácilmente. (Git, s.f.).

a. Sistema de control de versiones locales. - El a tratar este problema, los programadores desarrollaron hace mucho tiempo VCS locales que tenían una base de datos simple que mantenía todos los cambios en los archivos bajo control de revisión.

Una de las herramientas VCS más populares fue un sistema llamado RCS, que todavía se distribuye con muchas computadoras hoy en día. RCS funciona manteniendo conjuntos de parches (es decir, las diferencias entre archivos) en un formato especial en el disco; luego puede volver a crear el aspecto de cualquier archivo en cualquier momento agregando todos los parches.

b. Sistema centralizado de control de versiones. El sistema de control de versiones centralizados (CVCS). tienen un único servidor que contiene todos los archivos versionados y varios clientes que extraen archivos de ese lugar central. Los administradores tienen un control preciso sobre quién puede hacer qué, y es mucho más fácil administrar un CVCS que tratar con bases de datos locales en cada cliente.

2.16 JAVA

2.16.1 ¿Qué es Java?

Según (JAVA, s.f.), defina a Java como una tecnología que se usa para el desarrollo de aplicaciones que convierten a la Web en un elemento más interesante y útil., Java le permite jugar, cargar fotografías, chatear en línea, realizar visitas virtuales y utilizar servicios como, por ejemplo, cursos en línea, servicios bancarios en línea y mapas interactivos. Si no dispone de Java, muchas aplicaciones y sitios web no funcionarán. (JAVA, s.f.).

Java es la base para prácticamente todos los tipos de aplicaciones de red, además del estándar global para desarrollar y distribuir aplicaciones móviles y embebidas, juegos, contenido basado en web y software de empresa. Con más de 9 millones de desarrolladores en todo el mundo, Java le permite desarrollar, implementar y utilizar de forma eficaz interesantes aplicaciones y servicios..(*¿Qué es Java?*, s. f.)

2.16.2 ¿Qué Recibiré al Descargar Software de Java?

Según (JAVA, s.f.), Java Runtime Environment (JRE) es lo que se obtiene al descargar el software de Java. JRE está formado por Java Virtual Machine (JVM), clases del núcleo de la plataforma Java y bibliotecas de la plataforma Java de soporte. JRE es la parte de tiempo de ejecución del software de Java, que es todo lo que necesita para ejecutarlo en el explorador web.

2.16.3 ¿Por qué los Desarrolladores de Software Eligen Java?

Según (JAVA, s.f.), Java ha sido probado, ajustado, ampliado y probado por toda una comunidad de desarrolladores, arquitectos de aplicaciones y entusiastas de Java. Java está diseñado para permitir el desarrollo de aplicaciones portátiles de elevado rendimiento para el más amplio rango de plataformas informáticas posible. Al poner a disposición de todo el mundo aplicaciones en entornos heterogéneos, las empresas pueden proporcionar más servicios y mejorar la productividad, las comunicaciones y colaboración del usuario final y reducir drásticamente el costo de propiedad tanto para aplicaciones de usuario como de empresa.

Según (JAVA, s.f.), Java se ha convertido en la herramienta más importante para los desarrolladores, ya que les permite:

- Escribir software en una plataforma y ejecutarla virtualmente en otra
- Crear programas que se puedan ejecutar en un explorador y acceder a servicios Web disponibles.
- Desarrollar aplicaciones de servidor para foros en línea, almacenes, encuestas, procesamiento de formularios HTML y mucho más.
- Combinar aplicaciones o servicios que utilizan el lenguaje Java para crear aplicaciones o servicios con un gran nivel de personalización.
- Escribir aplicaciones potentes y eficaces para teléfonos móviles, procesadores

remotos, microcontroladores, módulos inalámbricos, sensores, gateways, productos de consumo y prácticamente cualquier otro dispositivo electrónico.

2.17 ¿Qué es NetBeans?

NetBeans es un entorno de desarrollo gratuito y de código abierto que permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o para dispositivos móviles. Contiene muchas funcionalidades, para distintos tipos de aplicaciones y para facilitar al máximo la programación, la prueba y la depuración de las aplicaciones que se desarrollan. También incorpora un editor propio. Da soporte a las siguientes tecnologías, entre otras: Java**,** PHP**,** Groovy**,** C/C++**,** HTML5, Además puede instalarse en varios sistemas operativos: Windows, Linux, Mac OS. (netbeans.apache.org, 2017).

2.17.1 Características Principales

Según (netbeans.apache.org, 2017), Netbeans presenta algunas características entre las que se destacan a continuación:

- Suele dar soporte a casi todas las novedades en el lenguaje Java.
- **Asistentes** para la creación y configuración de distintos proyectos, incluida la elección de algunos frameworks.
- **Editor de código**, **multilenguaje**, con el habitual coloreado y sugerencias de código, acceso a clases pinchando en el código, control de versiones,
- **Gestión de grandes proyectos.,** se simplifica con el uso de diferentes vistas, asistentes de ayuda, y estructurando la visualización de manera ordenada,
- **Herramienta para depurado de errores**: el debugger que incluye el IDE es bastante útil para encontrar dónde los errores con gran facilidad.
- **Optimización de código**: por su parte el Profiler nos ayuda a optimizar nuestras aplicaciones e intentar hacer que se ejecuten más rápido y con el mínimo uso de

memoria.

- **Acceso a base de datos**: desde el propio Netbeans podemos conectarnos a distintos sistemas gestores de bases de datos, como Oracle, MySql.
- **Servidores de aplicaciones**, Se integra con diversos servidores pudiendo gestionarlos desde el propio IDE.

2.18 Apache Maven

2.18.1 Qué es Apache Maven

Según (OpenWebinars, s.f.)**,** Apache Maven es una herramienta de gestión de proyectos que estandariza a traves del POM **(**Project Object Model) la configuración de un proyecto en todo su ciclo de vida**.** También se considera como un sistema de gestión de dependencias muy potente y una lógica de ejecución de plugins que permite, en determinadas las fases del ciclo de vida, poder ejecutar un plugin concreto que ofrezca una solución a un problema. (OpenWebinars, s.f.).

El nombre proviene de una palabra hebrea, que posteriormente fue acuñada en inglés como Maven, que significa “persona con conocimiento” o “profesor”. El concepto de Maven es acumular conocimiento y reutilización de código, estandarización y una buena definición de configuraciones de proyectos. (OpenWebinars, s.f.)

2.18.2 Principales Características de Apache Maven

Según (OpenWebinars, s.f.), Apache Maven ofrece algunas características principales ya que es la base d ellos copiladores actuales como Eclipse, NetBeans o IntelliJ entre otros, entre ellas tenemos:

- Un sistema de gestión dependencias.
- Un mecanismo distribuido de distribución de librerías. El comportamiento distribuido es siempre desde el repositorio local de Maven hacia los repositorios que están publicados en Internet o en la red corporativa.

- Mecanismos para ser extensible, por la creación de plugins.
- Es multi-plataforma, puede funcionar tanto en entornos Linux como Windows al ser una aplicación Java.
- Es software libre, con lo cual es el código está disponible,
- Fomenta la reutilización de código y de librerías. El hecho de que Apache Maven ofrezca repositorios oficiales y públicos de software libre, con librerías desplegadas, que toda la comunidad de desarrolladores de software utiliza
- Es compatible con múltiples IDEs.

2.19 Maven y la Gestión de Librerías

Maven se destaca en la gestión de las librerías, que son un conjunto de recursos, funciones, métodos, código ya implementado, que podemos reutilizar cuando desarrollamos software. Cuando programamos, tendemos a reutilizar código, recursos que nos ahorren tiempo y nos faciliten el desarrollo. Un ejemplo típico es que necesitemos un sistema de logs para registrar los fallos que puedan surgir al ejecutar la aplicación y poder solucionarlos. Esto es muy común en todos los desarrollos de software y no nos centramos en programar un sistema propio de logs sino que utilizamos una librería ya hecha, probada y que sea ocupada por los desarrolladores, otro ejemple es pueda que queramos hacer pruebas unitarias en este caso utilizaremos Junit y podemos incorporar esta librería de muestro desarrollo, todas estas librerías son creadas por los desarrolladores a nivel mundial. (GARZAS, 2014).

No obstante, puede que dentro de nuestra propia organización haya algún código que tengamos que utilizar en varios proyectos, y sea recomendable transformarlo en una librería propia de la organización. (GARZAS, 2014) en lugar de tener que estar buscando e instalando manualmente las librerías para nuestros proyectos, con Maven podemos indicar en el POM.xml qué librerías necesita nuestro

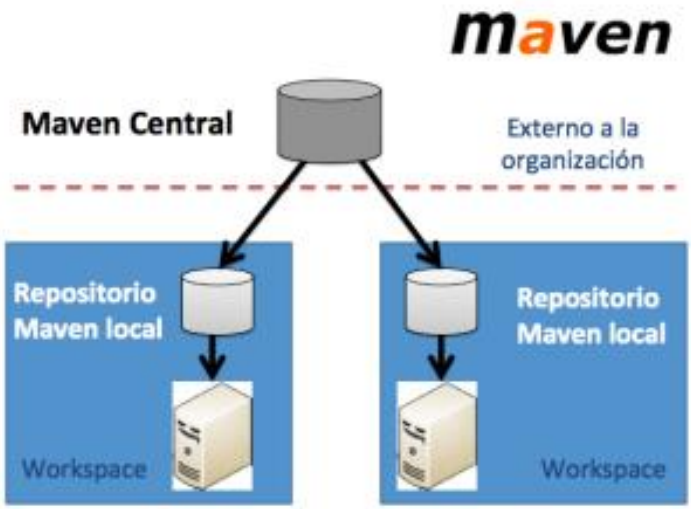
proyecto, y él se encargará de buscarlas para que podamos utilizar los métodos, recursos de esas librerías al programar, al compilar etc. (GARZAS, 2014).

Para entender cómo gestiona Maven las librerías o incluso cómo compila, es bueno ver cómo se organiza, cuál es su arquitectura. Maven se organiza a través de repositorios, donde van a estar esas librerías.

En el esquema más simple de Maven es que tendríamos un repositorio local en cada ordenador donde tengamos instalado Maven y un repositorio remoto y público, llamado Maven Central. Y es ahí donde están almacenadas casi todas las librerías que puedas necesitar a la hora de desarrollar software. (GARZAS, 2014).

Figura 4:

Gestión de librerías de Maven



Nota: La figura muestra como Maven gestiona las librerías y componentes. Imagen tomada desde (GARZAS 2014).

Según (GARZAS, 2014), a la hora de compilar el código o cuando queramos utilizar las librerías, Maven haría lo siguiente:

1. En el pom.xml del proyecto, en la sección de dependencias, indicamos las librerías que usa nuestro proyecto y que están alojadas en Maven central.

2. Cuando doy la orden de compilar el código o instalar las librerías, Maven buscará en mi repositorio local si están esas librerías. Si no están ahí, se irá a buscarlas al Maven central, las descargará e instalará en el repositorio local.

Luego compilará el código utilizando esas librerías de Maven central que están el repositorio local, o ya podremos usar los métodos que proporcionan en nuestro código, la siguiente vez que Maven compile, hará el mismo proceso, pero será más rápido, porque al tener esas librerías instaladas ya en el repositorio local, no le hará falta descargarlas.

2.20 ¿Qué es un Repositorio?

Según (Sonatype, 2008), un repositorio es la ubicación de almacenamiento centralizado donde se recuperan o almacenan componentes como paquetes, bibliotecas, librerías, binarios o contenedores para que se puedan instalar o usarse. Para facilitar el uso y consumo de componentes de código abierto, se agregan en colecciones de repositorios públicos y por lo general están en el internet como un servicio.

2.20.1 Tipos de Repositorios

En Nexus Repository Manager existen tres tipos de repositorios: repositorios proxy, repositorios alojados y grupos de repositorios, todos con diferentes formatos de repositorio que se detallan a continuación.

- **Repositorio Proxy. –** Es aquel que está vinculado a un repositorio remoto, cualquier solicitud de un componente se verifica con el contenido local del repositorio proxy. Si no se encuentra ningún componente local, la solicitud se reenvía al repositorio remoto. El componente se recupera y almacena localmente

en el administrador del repositorio, que actúa como caché. Cualquier solicitud futura para el mismo componente se cumple desde el almacenamiento local, eliminando el ancho de banda de la red y la sobrecarga de tiempo al recuperar el componente desde el repositorio remoto nuevamente. Además, cuando los componentes se descargan al proxy de Nexus Repositorio Manager, una copia permanece allí indefinidamente, a menos que sea eliminada por un administrador de Repositorio Manager. Esto es extremadamente útil en caso de que un componente no esté disponible en el repositorio remoto. (Sonatype, 2008).

- **Repositorios Hospedados. -** Un repositorio alojado es un repositorio que almacena componentes en el administrador de repositorios como la ubicación autorizada para estos componentes. Por ejemplo, cree un repositorio alojado cuando tenga componentes internos que no se descargan de un repositorio público, pero que son utilizados por varios equipos de desarrollo en su organización. (Sonatype, 2008).
- **Repositorios Grupales. -** Un grupo de repositorios es una colección de otros repositorios, donde puede combinar múltiples repositorios del mismo formato en un solo elemento. Esto representa una característica poderosa de Nexus Repository Manager que permite a los desarrolladores confiar en una única URL para sus necesidades de configuración. Por ejemplo, si su grupo tiene un repositorio proxy Maven Central y un repositorio alojado para JAR de terceros, estos se pueden combinar en un grupo con una URL para compilaciones. (Sonatype, 2008).

2.20.2 Gerente de Repositorio

Un administrador de repositorios es una aplicación de servidor dedicada, utilizado para administrar todos los repositorios que usan sus equipos de desarrollo en el transcurso del desarrollo. En conclusión, un administrador de repositorios hace lo siguiente: (Sonatype, 2008).

- Repositorios Proxies remotos y contenidos de caches,
- Repositorios hospedados internos,
- Agrupa los repositorios en un solo repositorio.

Un administrador de repositorios se convierte en la plataforma de almacenamiento central y autorizada para todos los elementos de código abierto y propietarios producidos por su equipo de desarrollo. Con un administrador de repositorios, puedo controlar completamente el acceso y la implementación de cada componente de su organización desde una única ubicación, permite controlar lo que ingresa a sus productos desde fuentes remotas, así como examinar y realizar un seguimiento de lo que producen sus sistemas de compilación. Además, el administrador de repositorios permite asegurar una conexión a repositorios remotos asegurando que su uso no esté expuesto públicamente. (Sonatype, 2008).

2.20.3 ¿Por qué Necesito un Administrador de Repositorio?

En la actualidad se ensambla software utilizando código abierto, propietario y código de terceros, es por esta razón muchas organizaciones confían en la gestión de repositorios para obtener, almacenar, compartir e implementar de manera eficiente partes de software. En lugar de descargar directamente de repositorios públicos, descargas repetidas o distribución manual, el uso de un administrador de repositorios lo convierte en el punto central de acceso y administración para el uso de cualquier componente en su ciclo de vida de desarrollo de software, esta función central facilita

que todos en su organización comprendan donde se almacenan las cosas. (Sonatype, 2008).

Dentro de su empresa el uso de un administrador de repositorios le permite administrar recursos necesarios para el desarrollo del software, la implementación de aplicaciones y el aprovisionamiento automatizado de hardware, cumple un papel central para la gestión del ciclo de vida de las aplicaciones, Un administrador de repositorios simplifica el mantenimiento de sus repositorios internos y el acceso a los repositorios remotos. (Sonatype, 2008).

Integrar un administrador a su empresa le permite lograr lo siguiente: (Sonatype, 2008).

- Administre código de software a través del SDCL.
- Buscar y navegar por componentes de software.
- Contralar los lanzamientos de software.
- Integrar con sistemas de seguridad externos.
- Ver metadatos de componentes.
- Alojar elementos internos no disponibles en repositorios remotos.
- Confiar en un administrador de repositorios le permite tener una nueva visión de la calidad de los elementos que fluyen a través del repositorio.

2.20.4 Componentes, Repositorios y Formatos de Repositorio

a. Componentes. Un componente es un recurso como una biblioteca o un marco que se utiliza como parte de su aplicación de software en tiempo de ejecución, integración o tiempo de ejecución de pruebas unitarias, o requerido como parte de su proceso de compilación. Podría ser una aplicación completa o un recurso estático como una imagen. Por lo general, estos componentes son archivos de una gran variedad de archivos que incluyen. (Sonatype, 2008).

- Código de bytes de Java en archivos de clase.
- Archivos de texto, por ejemplo, archivos de propiedades, archivos XML, código JavaScript, HTML, CSS.
- Archivos binarios como imágenes, archivos PDF, archivos de sonido

Según (Sonatype, 2008), los archivos están utilizando numerosos formatos. como:

- Java JAR, WAR, formatos EAR.
- Archivos ZIP o .tar.gz simples.
- Atros formatos de paquete como paquetes NuGet, gemas Ruby, paquetes NPM
- Formatos ejecutables como archivos .exe o.sharchivos Android APK, varios formatos de instalador.

Los componentes pueden estar compuestos por múltiples componentes anidados. Por ejemplo, considere una aplicación web Java empaquetada como un componente WAR. Contiene varios componentes JAR y varias bibliotecas de JavaScript. Todos estos son componentes independientes en otros contextos y se incluyen como parte del componente WAR. (Sonatype, 2008).

Los componentes proporcionan todos los componentes básicos y las características que permiten que un equipo de desarrollo cree aplicaciones potentes ensamblándolas y agregando sus propios componentes relacionados con el negocio para crear una aplicación potente y completa. (Sonatype, 2008).

En diferentes cadenas de herramientas, los componentes se denominan artefacto, paquete, paquete, archivo y otros términos. El concepto y la idea siguen siendo los mismos y el componente se utiliza como término genérico. (Sonatype, 2008)

Los componentes se identifican mediante un conjunto de valores específicos: las coordenadas. Un conjunto genérico de estas coordenadas es el uso de grupo, nombre y

versión. Los nombres y el uso de estas coordenadas cambian con las cadenas de herramientas utilizadas. Los componentes también pueden ser el ancla para otros metadatos. (Sonatype, 2008).

2.20.5 Componentes de un Repositorio

Existe una amplia variedad de componentes y la comunidad de código abierto. Se ha convertido en un patrón predeterminado para crear aplicaciones combinando las características de múltiples componentes con sus propios componentes personalizados que contienen el código de su aplicación para crear una aplicación para un dominio específico. (Sonatype, 2008).

Para facilitar el consumo y el uso de componentes, se agregan en colecciones de componentes. Estos se denominan repositorio y generalmente están disponibles en Internet como un servicio. En diferentes plataformas se utilizan términos como registro y otros para el mismo concepto. (Sonatype, 2008).

a. **Formatos de repositorio. –** Los repositorios usan diferentes tecnologías para almacenar y exponer los componentes en ellos a las herramientas del cliente. Esto define un formato de repositorio y está estrechamente relacionado con las herramientas que interactúan con el repositorio. Por citar un ejemplo el formato del repositorio de Maven se basa en una estructura de directorio específica que viene definida por los identificadores de los componentes y una cantidad de archivos con formato XML para metadatos. Otros formatos de repositorio utilizan bases de datos para el almacenamiento y las interacciones, o diferentes estructuras de directorios con archivos específicos de formato para los metadatos. (Sonatype, 2008).

2.21 El Administrador De Repositorios De Nexus

2.21.1 ¿Qué es Nexus?

Nexus una relación o conexión entre personas o cosas. Nexus es un marco de trabajo que consiste en roles, eventos, artefactos y técnicas que vinculan y entrelazan el trabajo en una sola Lista de Producto para construir un Incremento Integrado que cumpla un objetivo.

2.21.2 Fundamentos de Nexus

El desarrollo de software es complejo y la integración de ese trabajo en software funcional tiene muchos artefactos y actividades que deben coordinarse para crear un resultado “Terminado”. El trabajo debe organizarse, secuenciarse, las dependencias deben resolverse y los resultados deben obtenerse por etapas.

Muchos desarrolladores de software han usado el marco de trabajo Scrum para trabajar colectivamente como equipo con el fin de desarrollar un Incremento de software funcional. Sin embargo, si más de un Equipo Scrum está trabajando sobre la misma Lista de Producto y la misma base de código para un producto, se presentan dificultades. Si los desarrolladores no están situados en el mismo lugar, ¿cómo se comunicarán cuando estén realizando un trabajo que los afecte entre sí? Si trabajan en equipos diferentes, ¿cómo integrarán su trabajo y probarán el Incremento Integrado? Estos desafíos aparecen cuando dos equipos se están integrando y se vuelve significativamente más difícil cuando tres o más equipos Scrum integran su trabajo en un único incremento.

Muchas dependencias surgen durante el trabajo de múltiples equipos que colaboran para crear al menos un Incremento completo y “Terminado” en cada Sprint. Estas dependencias se relacionan con:

a. Requisitos: El alcance de los requisitos puede traslaparse y la manera en la que se implementan también puede afectar a los demás. Ese conocimiento debería tenerse en cuenta al ordenar la Lista de Producto y seleccionar los requisitos.

b. Conocimiento del dominio: Las personas en los equipos tienen conocimiento de varios sistemas de negocio y computacionales. Ese conocimiento debería mapearse a los Equipos Scrum para asegurar que es suficiente y también para minimizar las interrupciones entre equipos durante el Sprint.

c. Artefactos de software y de pruebas: Los requisitos son o serán instanciados en código de software.

En la medida que los requisitos, el conocimiento de los miembros de los equipos y los artefactos de código/pruebas se mapeen a los mismos Equipos Scrum, las dependencias entre los equipos se pueden reducir.

Cuando la entrega de software usando Scrum se escala, estas dependencias de requisitos, conocimiento del dominio y artefactos de software deberían orientar la organización de los Equipos de Desarrollo. En la medida en que esto se haga, la productividad se optimizará.

2.21.3 ¿Cómo Gestiona Nexus Repository los Componentes?

Con Nexus Repository Manager, puede administrar componentes desde el desarrollo hasta la entrega, incluidos archivos binarios, contenedores, ensamblajes y aplicaciones terminadas. En un administrador de repositorio, los componentes se identifican mediante un conjunto de valores específicos definidos por el formato. Por ejemplo, los componentes de Maven usan un conjunto genérico de valores de coordenadas llamados grupo, ID de artefacto y versión. (Sonatype, 2008).

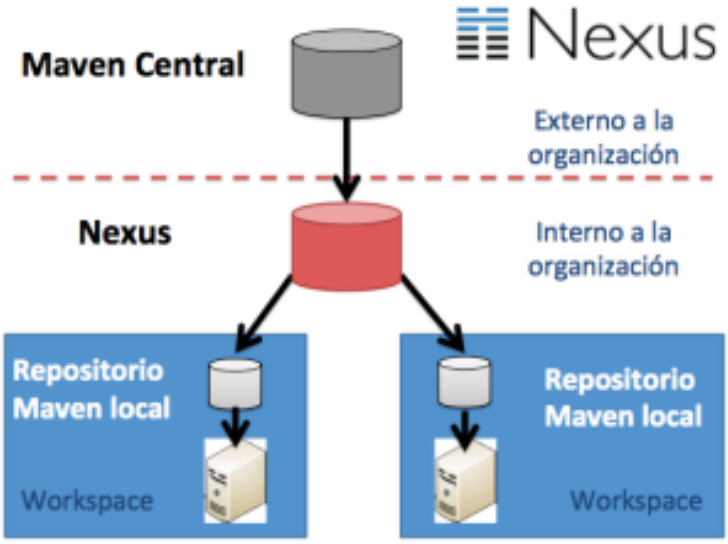
Nexus Repository Manager proporciona búsquedas integradas que le permiten localizar e inspeccionar fácilmente los detalles de los componentes. La búsqueda de

componentes descargados en el administrador de repositorios le permite acceder a detalles específicos, como las diferentes versiones disponibles y los datos de la licencia. Esta información es esencial para las migraciones de herramientas de compilación, la descarga de paquetes de implementación y otras actividades de desarrollo y operaciones relacionadas con los componentes. (Sonatype, 2008).

Como podemos ver el trabajo que desarrolla Maven central y los desarrolladores, esto ya es muy útil y nos ahorra mucho tiempo y preocupaciones. Ya pensando en una empresa, lo cierto es que no nos interesa estar dependiendo siempre de algo externo a la organización y sobre todo, si queremos utilizar librerías propias de la empresa y al instalar el Administrador de Repositorio de Nexus el esquema anterior pasaría a ser el siguiente:

Figura 5:

Proceso de gestión de repositorios con Nexus



Nota:La figura muestra como Nexus gestiona las librerías y componentes. Imagen tomada desde (GARZAS 2014).

En este caso, al compilar o usar las librerías, Maven primero buscará en el repositorio local, si no están ahí buscará en el Nexus y si no se irá a Maven Central y las instalará en el Nexus. Las librerías propias de la empresa las subiremos directamente al Nexus. Este tipo de repositorios también se utilizan como cachés. La idea de tener un Nexus, es que sean repositorios de librerías internos en la organización. Al estar en una red local, el acceso al Nexus será más rápido que tener que conectarme a Maven Central para descargar las librerías.

Además, ese Nexus es compartido por toda la organización, por todo el equipo de desarrollo, si alguien del equipo tiene que compilar el código, y ciertas librerías de Maven Central no están en el Nexus, su Maven hará que se descarguen e instalen esas librerías en Nexus.

Cuando otra persona vuelva a compilar el código, ya no tendrá que bajarse las librerías de Maven Central, porque el Maven de otro desarrollador ya lo hizo. Así también ahorramos tiempo.

Y otra cosa muy útil, si mi proyecto tiene muchas dependencias internas, es decir, cuando compilo este módulo se genera un artefacto que utilizo como base de la compilación de otro módulo etc. Estos artefactos intermedios se guardarán en Nexus, permitiendo así, que otro desarrollador al compilar no tenga que volver a generarlos y su Maven pueda cogerlos de ahí.

CAPÍTULO III

DESARROLLO

3.1 Administrador Nexus Repository Manager (NXRM)

Si tu empresa al desarrollar software sin un administrador de repositorios, está perdiendo una serie de beneficios, tiempo, recursos económicos y oportunidades de reducir posibles errores, si todos los miembros utilizan los repositorios públicos como el repositorio central para descargar componentes pierde ganancias en velocidad y eficiencia. Es hora de empezar a utilizar al máximo los recursos tecnológicos y comenzar a utilizar un administrador de repositorio NEXUS, a continuación, he recopilado una serie de temas, detallando paso a paso como descargar, instalar, administrar el Administrador de repositorios NEXUS. (Sonatype, 2008).

En el siguiente apartado voy a ir detallando paso a paso la instalación del Administrador de Repositorios de Nexus.

3.1.1 Descargar el Administrador de Repositorios de Nexus

Para descargar la última distribución OSS de Nexus Repository Manager, se debe dirigir a la página oficial de descarga de OSS de Sonatype y elija el archivo de paquete comprimido que se adapte a sus necesidades en la sección Nexus Repository Manager OSS 2.x. Comenzará la descarga de la última versión. (Sonatype, 2008).

3.1.2 Instalación

La instalación del repositorio es sencilla. Debe descomprimir el archivo de la aplicación web en un directorio, puede instalarlo en su directorio personal o donde desee, Nexus no tiene directorios codificados. Se ejecutará desde cualquier directorio. (Sonatype, 2008).

Si descargó el ZIP, use: descomprimir nexus-2.14.5-02-bundle.zip- 2.14. 5 - 02 - paquete . Código PostalY, si descarga el archivo TAR de GZip'd, ejecute:

tar xvzf nexus-2.14.5-02-bundle.tar.gz- 2.14 . 5 - 02 - paquete . alquitrán . gz

Si está instalando el administrador de repositorio en un servidor en una máquina Unix, este libro asume que está instalado /usr/local/nexus-2.14.4-03 con un enlace simbólico /usr/local/nexus al directorio nexus y en Windows, debe instalar el administrador de repositorio fuera de Archivos de programa para evitar problemas con la virtualización del registro de archivos de Windows, puede instalarlo en el directorio AppData \ Local del directorio de inicio de ese usuario. (Sonatype, 2008).

3.1.3 Actualización

Es muy fácil actualizar una instalación existente, se debe descomprimir el archivo en el directorio que contiene la instalación existente. Una vez que se descomprime el archivo, el nuevo directorio detenga el servidor y cámbielo para que apunte al nuevo directorio de la aplicación.

3.1.4 Levantar el Servicio

Cuando inicia el administrador de repositorio, está iniciando un servidor web en el puerto predeterminado 0.0.0.0: 8081. Se ejecuta dentro de un contenedor de servicios y se puede configurar para ejecutar el administrador de repositorio como un servicio de Windows o un demonio de Unix. (Sonatype, 2008).

Para iniciar el administrador de repositorio en una plataforma similar a Unix como Linux, MacOSX o Solaris, use:

cd / usr / local / nexus

./ bin / nexus consola

En los sistemas Unix, puede iniciar el administrador de repositorios separado del shell de inicio con el comando de inicio incluso cuando aún no está instalado como servicio.

. / bin / nexus start

Cuando se ejecute, debería ver un mensaje de retroalimentación y luego puede seguir el proceso de inicio viendo los logs/wrapper.log cambios del archivo de registro.

Iniciando Nexus Repository Manager ... Iniciado Nexus Repository Manager.

$ tail - f logs / envoltorio. Iniciar sesión

El administrador del repositorio se ejecutará y escuchará en todas las direcciones IP (0.0.0.0) que están configuradas para el host actual en el puerto 8081. Y de esta manera podrá utilizar la interfaz de usuario. (Sonatype, 2008).

Figura 6:

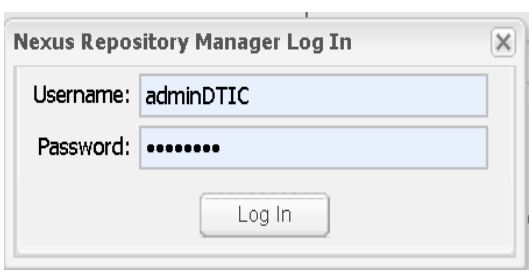
Servicio de levantado de Nexus

Nota: la figura muestra cómo se levanta un servicio en Nexus a través de un servidor. Elaborada por autor.

Haga clic en el enlace Iniciar sesión en la esquina superior derecha de la página web y debería ver el cuadro de diálogo de inicio de sesión donde el nombre del usuario es *admin y la contraseña es admin123.*

Figura 7:

Inicio de sección de Nexus



Nota: La imagen muestra el inicio de sección. Elaborada por autor.

3.1.5 Lista de Verificación Posterior a la Instalación

Según (Sonatype, 2008), posterior a la instalación se debe verificar la instalación del administrador de repositorios de Nexus completando las tareas que detallo a continuación:

- Cambie la contraseña administrativa y la dirección de correo electrónico. - Ya que viene por defectos el usuario: admin y la contraseña: admin123 y deben cambiarse por seguridad.
- Configurar los ajustes SMTP. – Utilizado para enviar correos electrónicos de recuperación de nombre de usuario y contraseña.
- Configurar los ajustes de proxy HTTP y HTTPS predeterminados. – Cuando necesite utilizar como proxy, solo se puede acceder a través de un servidor proxy HTTP o HTTPS interno de la empresa.
- Habilite las descargas de índices remotos. – NRMX viene con dos importantes repositorios proxy para el repositorio Maven Central y el repositorio Apache Snapshot.
- Cambiar la contraseña de implementación. - Esta predeterminada en *deployment123* para asegurar que solo desarrolladores autorizados pueden implementar componentes.
- Configurar las variables de entorno LANG. - Útiles para almacenar la configuración y los datos utilizando un juego de caracteres internacional
- Configurar rutas. - Define los patrones utilizados para definir e identificar los repositorios en los que se buscan los componentes

3.2 Configurar como Servicio al Administrador de Repositorio de Nexus.

Al instalar el Administrador de Repositorio de Nexus para uso de producción, debe configurarlo para que se ejecute como un servicio, de modo que se inicie una copia de seguridad después de reiniciar el servidor. (Sonatype, 2008).

3.2.1 Ejecución como Servicio en Linux

De acuerdo a (Sonatype, 2008), puede configurar el administrador del repositorio para que se inicie automáticamente copiando el script *nexus* en el */etc/init.d* directorio. En un sistema Linux, realice las siguientes operaciones como usuario root:

Cree un nexus usuario con suficientes derechos de acceso para ejecutar el servicio. Copiar $NEXUS_HOME/bin/nexus a /etc/init.d/nexus

Haga que el script /etc/init.d/nexus sea ejecutable y sea propiedad del usuario root -

chmod 755 / etc / init . d / nexus

chown root / etc / init . d / nexus

Edite este script cambiando las siguientes variables:

Cambiar NEXUS_HOME a la ubicación de la carpeta absoluta (por ejemplo, NEXUS_HOME="/usr/local/nexus")

Configure el RUN_AS_USER para nexus o cualquier otro usuario con derechos restringidos que desee utilizar para ejecutar el servicio. No debería ejecutar el administrador del repositorio como root.

Cambie PIDDIR a un directorio donde este usuario tenga permisos de lectura / escritura. En la mayoría de las distribuciones de Linux, /var/run solo se puede escribir por root.

Cambie el propietario y el grupo de los directorios utilizados por el administrador del repositorio.

Si Java no está en la ruta predeterminada para el usuario que ejecuta el administrador de repositorio, agregue una variable JAVA_HOME que apunte a su instalación de Java local y agregue $ JAVA_HOME / bin a la RUTA.

3.2.2 Ejecutar como Servicio de Windows

El script de inicio para el administrador de repositorio en plataformas Windows es bin/nexus.bat. los comandos para iniciar y detener el servicio star y stop y tiene los comandos adicionales install y uninstall. Una vez instalado como servicio con el comando de instalación, el archivo por lotes se puede utilizar para iniciar y detener el servicio. (Sonatype, 2008).

3.3 Maven y otras Herramientas de Construcción.

Históricamente, el administrador de Repositorios de Nexus comenzó como un administrador de repositorio compatible con el formato de repositorio de Maven y continúa fortaleciéndose incluyendo un excelente soporte para los usuarios de Apache Maven, Apache Ant / Ivy, Eclipse Aether, Gradle y o.tros. (Sonatype, 2008).

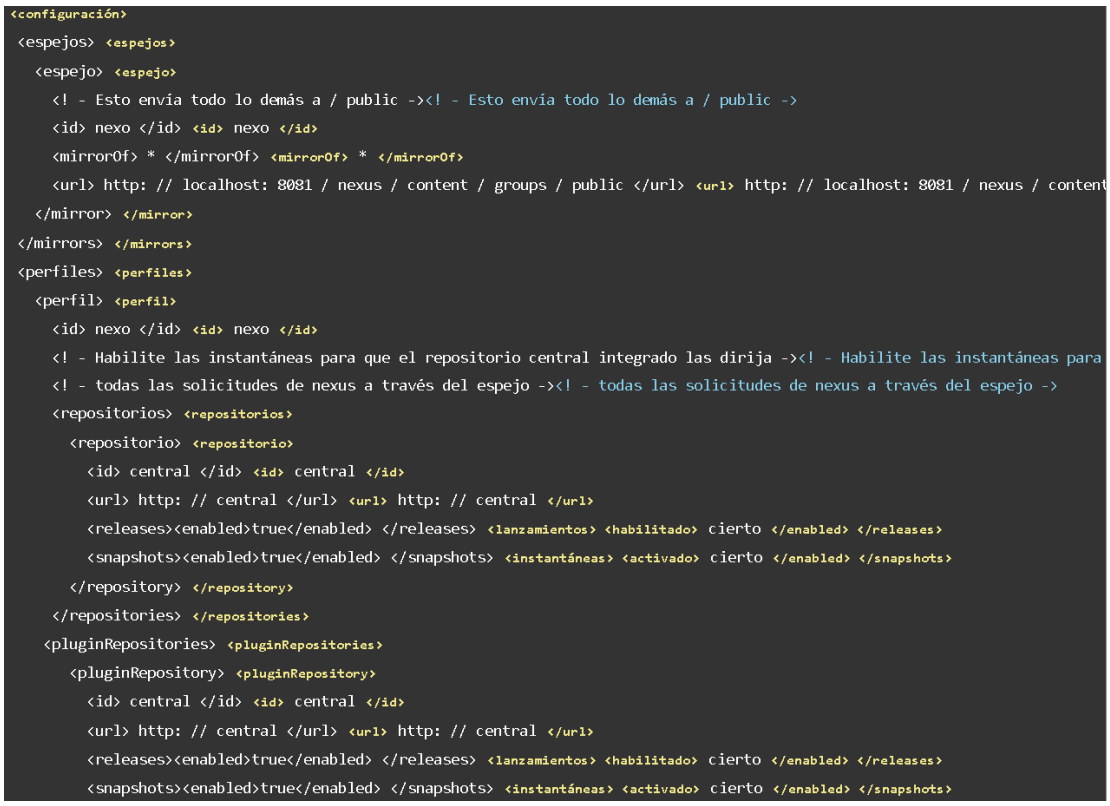
En este apartado se explica la configuración predeterminada incluida en el administrador de Repositorios de Nexus, instrucciones para crear más repositorios Maven, así como buscar y navegar por los repositorios.

3.3.1 Apache Maven

Para usar el administrador de Repositorios de Nexus con Apache Maven, configure Maven para verificar el administrador del repositorio en lugar de la conexión predeterminada e incorporada al Repositorio Central. Para hacer esto, agrega una configuración espejo y anula la configuración predeterminada para el repositorio central en su ~/.m2/settings.xml como se muestra. (Sonatype, 2008).

Figura 8:

Configuración se settings.xml en la carpeta .m2



Nota: La imagen muestra como las configuraciones en la carpeta .m2. Elaborada por autor.

En este ejemplo, Nexus se define un solo perfil llamado, sentting.xml configura el repositorio y los plugin del repositorio con el ID central que anula los mismos repositorios en el super POM.XML. El super POM.XML es interno a cada instalación de Apache Maven y establece valores predeterminados. Estas anulaciones son importantes ya que cambian los repositorios habilitando instantáneas y reemplazando la URL con una URL falsa. Esta URL es anulada por el configuración espejo en el mismo settings.xml archivo para apuntar a la URL de su grupo de repositorio único. Este grupo de repositorio puede, por lo tanto, contener tanto componente de lanzamiento como instantáneas y Maven los recogerá. (Sonatype, 2008).

3.3.2 Agregar Repositorios para las Dependencias Faltantes

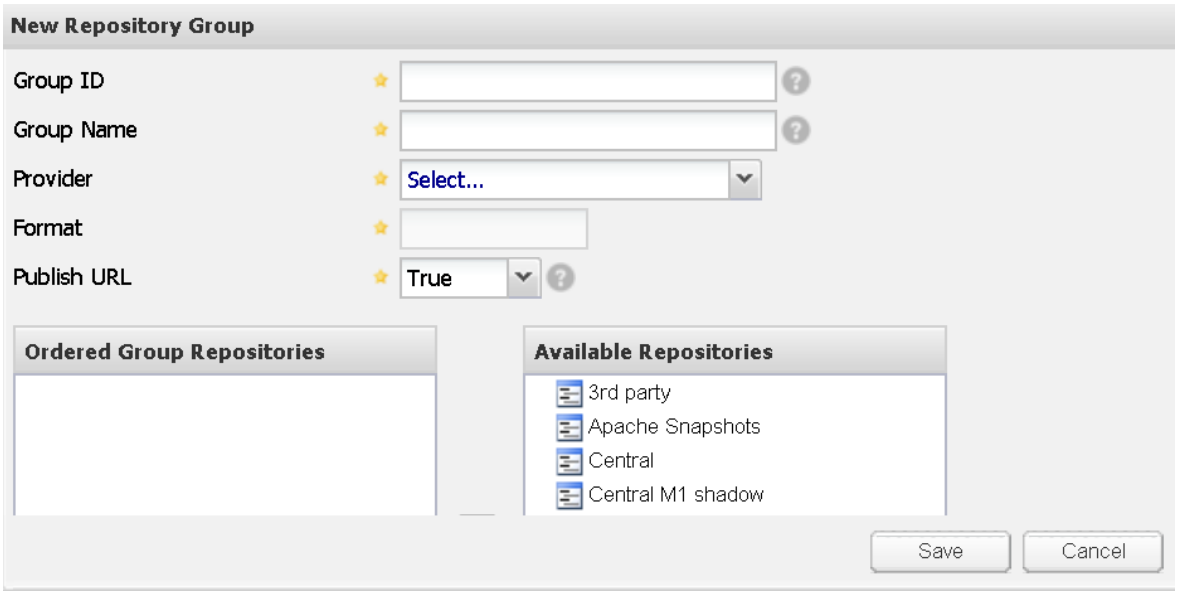
Si ha configurado su Maven settings.xml u otra configuración de herramienta de compilación para usar el grupo de repositorio público como un espejo para todos los repositorios, es posible que encuentre proyectos que no puedan recuperar componentes de la instalación del administrador de repositorio local. Es muy común que suceda esto porque está intentando crear un proyecto que ha definido un conjunto personalizado de repositorios y repositorios de instantáneas o se basa en el contenido de otros repositorios disponibles públicamente en su configuración. Cuando se encuentra con un proyecto de este tipo, todo lo que tiene que hacer es agregar este repositorio a su administrador de repositorio como un nuevo repositorio de proxy y luego agregar el nuevo repositorio de proxy al grupo público. (Sonatype, 2008).

3.3.3 Agregar un Repositorio a un Grupo

Después de agregar un nuevo repositorio, deberá agregar los nuevos repositorios al grupo de repositorios públicos. Para esta vamos a la opción de repositorio crear nuevo y escogemos la opción grupo de repositorios y procedemos a llenar los campos para poder crear un repositorio en grupos. (Sonatype, 2008).

Figura 9:

Agregar un Repositorio en grupo en Nexus



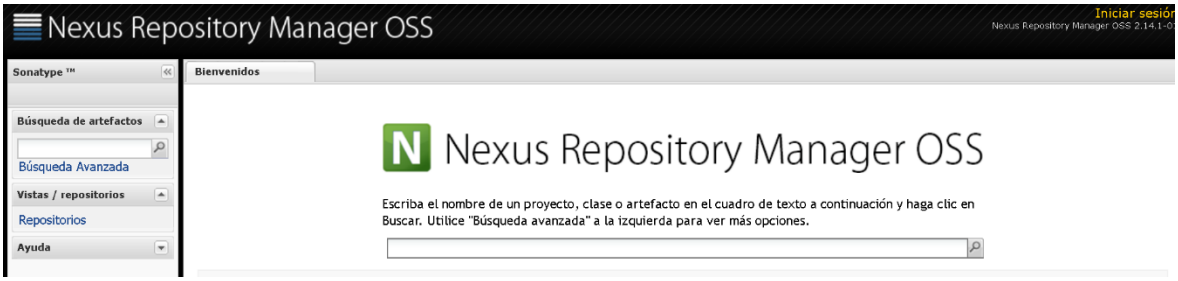
Nota: La imagen muestra cómo se agrega un grupo de repositorios. Elaborada por autor.

3.4 Interfaz Del Usuario

El administrador de repositorio de Nexus nos brinda acceso anónimo para los usuarios que solo necesitan buscar repositorios, explorar repositorios y examinar las fuentes del sistema. (Sonatype, 2008).

Figura 10:

Interfaz de usuario para usuario anónimos



Nota: La imagen muestra la interfaz de un usuario anónimo. Elaborada por autor.

La interfaz de usuario se utiliza con un navegador web y funciona mejor con los navegadores modernos. Y nos permite iniciar sección ingresando usuario y su contraseña. (Sonatype, 2008).

Figura 21:

Interfaz del usuario tipo administrador

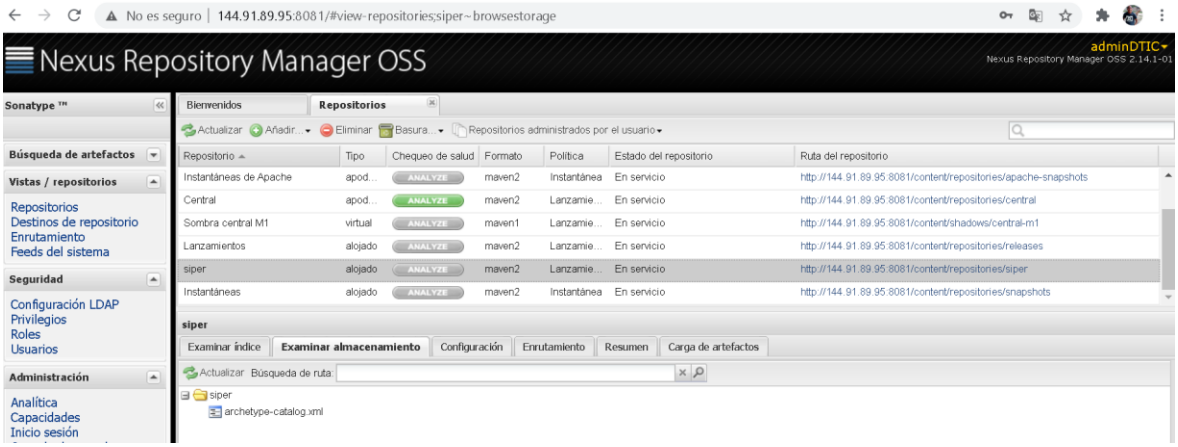
Nota: La imagen muestra la interfaz de un usuario tipo administrador. Elaborada por autor.

De acuerdo con la página oficial de (Sonatype, 2008), la interfaz de usuario se divide en varias secciones diferentes que voy a ir detallando a continuación.

- **Encabezamiento. -** La parte superior de la página contiene el encabezado y en el lado derecho el botón *Iniciar sesión*
- **Menú principal. -** El lado izquierdo de la aplicación se despliega el menú principal con varios submenús.
- **Panel principal.** -El panel principal de la aplicación a la derecha del menú principal puede albergar diferentes pestañas para diferentes selecciones en los submenús de la navegación.

Figura 32:

Interfaz del usuario administrador con sus respectivos repositorios.



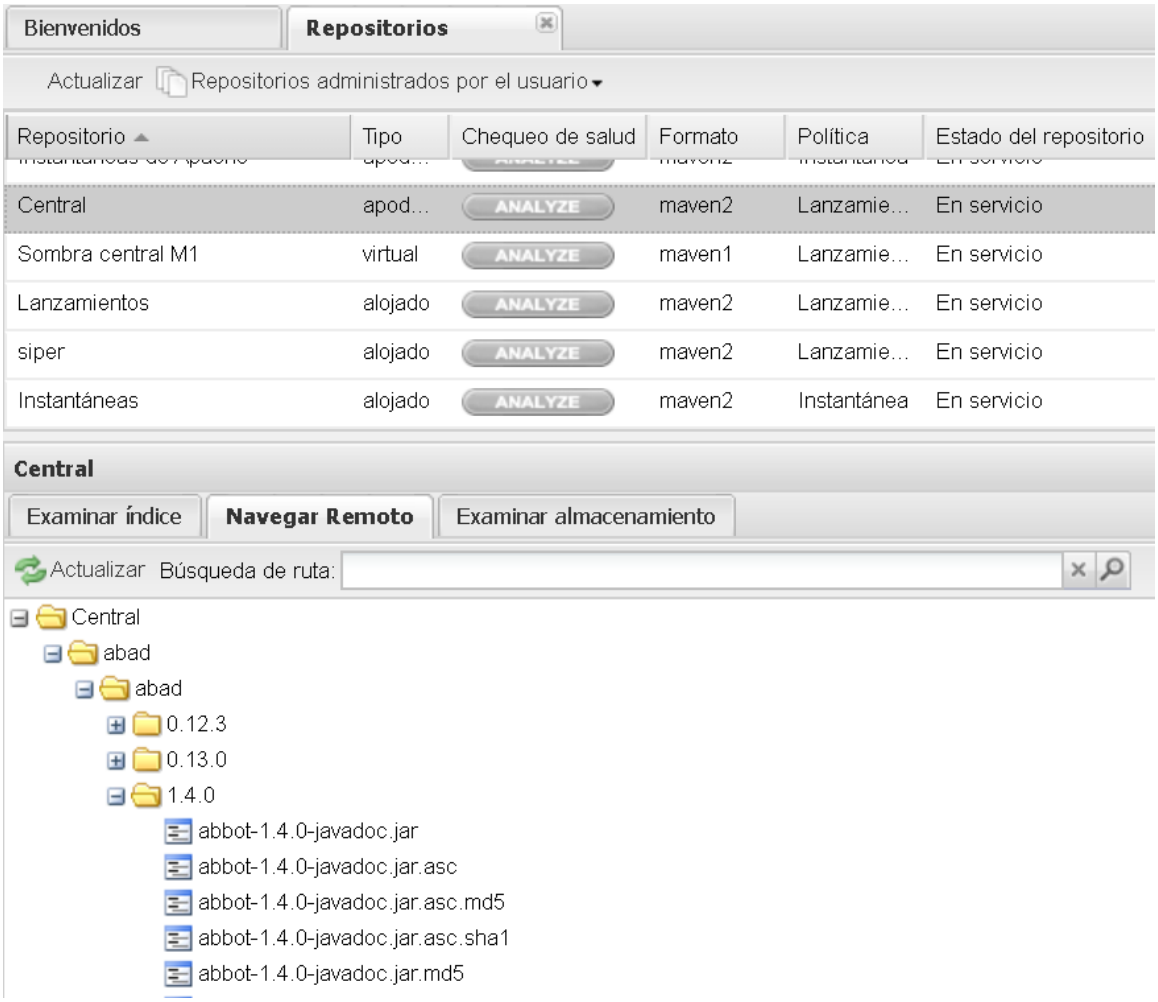
Nota: La imagen muestra la interfaz de un usuario administrador con sus respectivos repositorios. Elaborada por autor.

3.5 Navegación de Repositorios

Para examinar la estructura del administrador de repositorios y poder ver la estructura de cómo se encuentra organizado un repositorio hacemos clic en el menú vistas/ repositorios y a continuación se nos va desplegar las carpetas que contiene dicho repositorio como lo podemos ver en la siguiente figura. (Sonatype, 2008).

Figura 43:

Navegación de un repositorio mostrando sus distintos características



Nota: La imagen muestra a un repositorio desplegado sus diferentes características. Elaborada por autor.

3.6 Gestión de Repositorios.

Para administrar repositorios, se debe iniciar sección como usuario administrativo luego clic en repositorios en el menú vistas/ repositorios. El administrador de Repositorios de Nexus proporciona tres tipos de repositorios que son repositorios proxy, repositorios alojados y grupo de repositorios a continuación voy a detallar como se administra cada uno de ellos. (Sonatype, 2008).

3.6.1 Repositorios Proxy

Un repositorio proxy es un repositorio que está vinculado a un repositorio remoto, como el Repositorio Central. Cuando se usa un proxy, las solicitudes de componentes se verifican con los componentes almacenados en caché en su repositorio de proxy. Cuando busca componentes, si la solicitud no se encuentra en su repositorio proxy, se reenvía al repositorio central. Luego, ese componente se recupera de Central y se almacena en caché en el administrador del repositorio. Si busca ese mismo componente nuevamente, se encontrará en el almacenamiento local. Esto elimina la necesidad de ir al repositorio central y reduce el ancho de banda y el tiempo necesario para recuperar los componentes que necesita. Cuando los componentes se almacenan en un proxy de almacenamiento en caché, una copia de esos componentes permanece indefinidamente. En el caso de que un componente no esté disponible en el Repositorio central, aún tendrá acceso a él en su proxy. Esto proporciona más control sobre los componentes que necesita para crear sus aplicaciones. Nexus Repository Manager viene listo para usar con un repositorio proxy central de Maven ya configurado para acceder al repositorio central. (Sonatype, 2008).

Para crear un repositorio proxy se debe dar clic en repositorio luego añadir-repositorio proxy luego deberá llenar los campos obligatorios para poder crear un repositorio proxy.

Figura 54:

Creación de un repositorio Proxy en Nexus

Nota: La imagen muestra la creación de un repositorio proxy. Elaborada por autor.

3.6.2 Repositorio Alojado

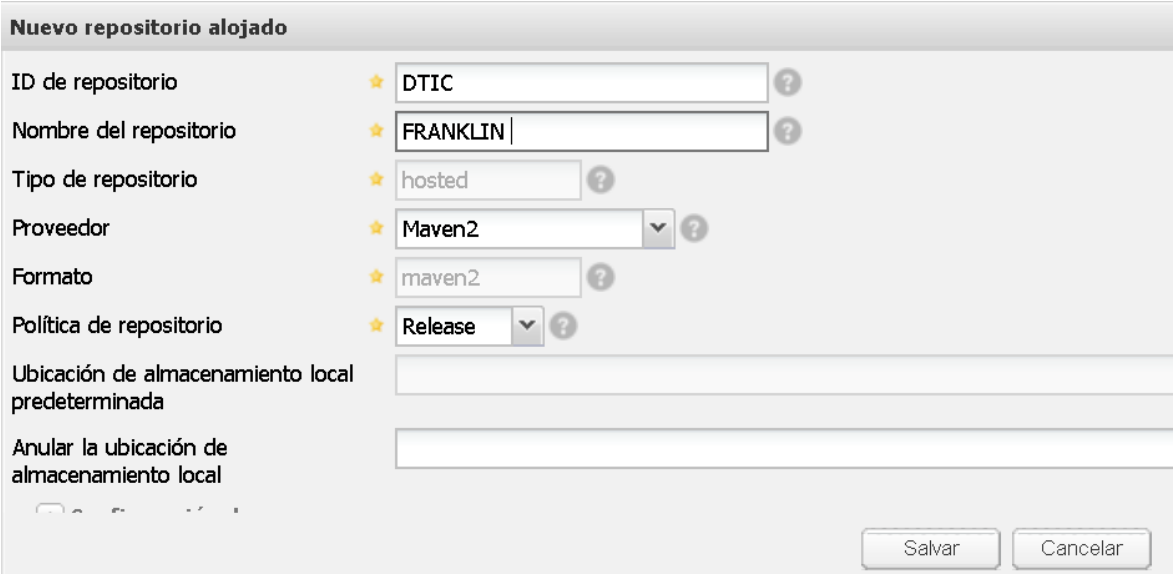
Este repositorio alojado es donde su organización puede publicar instantáneas internas de Maven. El Administrador de repositorios de Nexus incluye dos repositorios Maven alojados. El repositorio de maven-releases usa una política de versión de lanzamiento y el repositorio de maven-snapshots usa una política de versión de instantánea. Los repositorios de versiones están destinados a ser el lugar donde su organización publica versiones internas. Un componente de versión es un componente creado por una versión con versión específica. Estos componentes se consideran sólidos, estables y perpetuos para garantizar que las compilaciones que los utilizan sean repetibles en el tiempo. También puede utilizar este repositorio para componentes de terceros que no están disponibles en repositorios externos y no se pueden recuperar mediante un repositorio proxy configurado. Los componentes de las instantáneas cambian con el tiempo y se generan durante el desarrollo activo de un proyecto de software. El desarrollo continuo se realiza normalmente con versiones de instantáneas compatibles con la política de versión de instantáneas. (Sonatype, 2008).

Al igual que un entorno de producción y un entorno de prueba, recomendamos utilizar el repositorio de versiones para componentes estables en producción y un

repositorio de instantáneas para componentes que aún se encuentran en la fase de desarrollo. Para crear un repositorio alojado se debe dar clic en repositorio luego añadir-repositorio proxy luego deberá llenar los campos obligatorios para poder crear un repositorio proxy. (Sonatype, 2008).

Figura 65:

Creación de un repositorio hospedado



Nota: La imagen muestra la creación de un repositorio hospedado. Elaborada por autor.

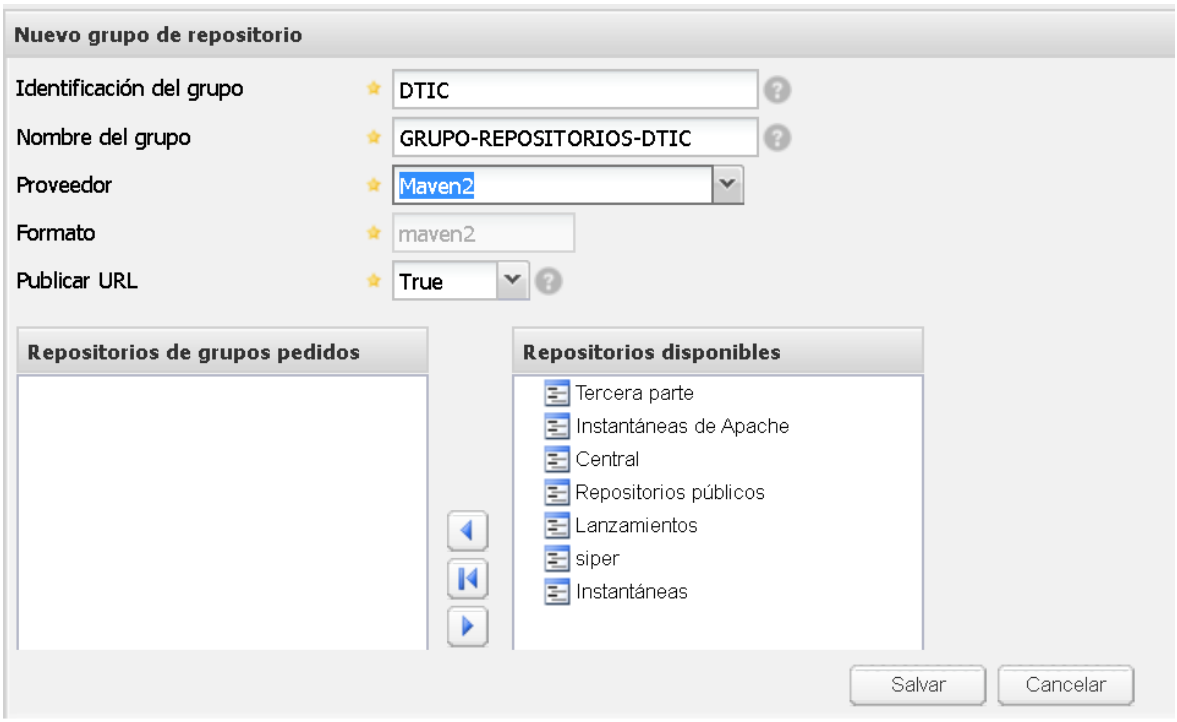
3.6.3 Los Grupos de Repositorios

Los grupos de repositorios son una característica poderosa del administrador de Repositorios de Nexus ya que permiten combinar varios repositorios y otros grupos de repositorios del mismo formato de repositorio en un solo grupo de repositorios. Este único grupo y la URL asociada se pueden utilizar como un único punto de acceso a todos los componentes en un formato específico procedente de varios repositorios. Esto facilita la configuración para los usuarios y al mismo tiempo permite a los administradores agregar más repositorios y por lo tanto componentes sin requerir cambios en los equipos cliente. (Sonatype, 2008).

Para crear un grupo de repositorios en el menú *Repositorios del* panel izquierdo en el menú *Vistas / Repositorios* haga clic y llene los campos necesarios para poder crear un grupo de repositorios.

Figura 76:

Creación de un grupo de repositorios



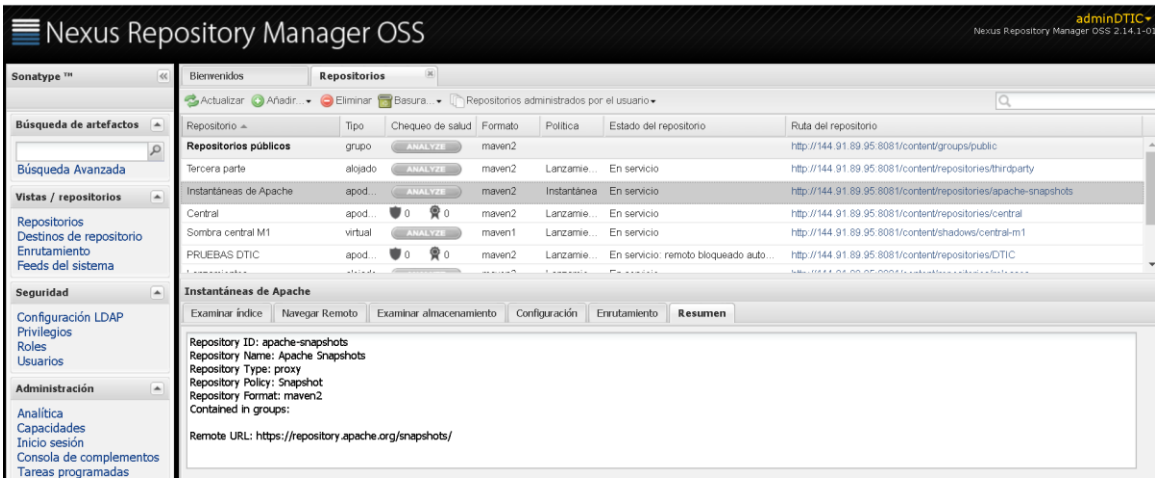
Nota: La imagen muestra la creación de un repositorio en grupo. Elaborada por autor.

3.6.4 Visualización del Panel de Resumen de un Repositorio

El panel Resumen se puede cargar seleccionando un repositorio alojado, proxy o virtual y luego haciendo clic en la pestaña Resumen.

Figura 87:

Visualización de un repositorio en general



Nota: La imagen muestra la pestaña de un repositorio en general. Elaborada por autor.

3.7 Gestionar la Seguridad

El administrador de Repositorios de Nexus utiliza un sistema de control de acceso basado en roles (RBAC) que brinda a los administradores un control muy detallado sobre quién puede leer desde un repositorio, quién puede administrar el servidor y quién puede implementar en los repositorios. El modelo de seguridad en el administrador del repositorio también es tan flexible que le permite especificar que solo ciertos usuarios o roles pueden implementar y administrar componentes en un repositorio específico bajo un groupId o clase de activo específico. (Sonatype, 2008).

Según la página oficial de (Sonatype, 2008), el sistema RBAC está diseñado en torno a los siguientes cuatro conceptos de seguridad:

- **Privilegios. -** Los privilegios controlan el acceso a una funcionalidad específica a cualquiera de los repositorios instalados o configurados. Los privilegios son derechos para leer, actualizar, crear o administrar recursos y realizar operaciones. El administrador del repositorio se envía con un conjunto de privilegios básicos que no se pueden modificar, y puede crear nuevos privilegios

para permitir una orientación detallada de los permisos de rol y usuario para repositorios específicos.

- **Objetivos**. - Los privilegios suelen estar asociados con recursos u objetivos. Un destino puede ser un repositorio específico o un conjunto de repositorios agrupados en algo llamado destino del repositorio.
- **Roles.** - Las colecciones de privilegios se pueden agrupar en roles para facilitar la definición de colecciones de privilegios comunes a determinadas clases de usuarios.
- **Usuarios**. - A los usuarios se les pueden asignar roles y modelarán a las personas que iniciarán sesión en el administrador de repositorios y leerán, implementarán o administrarán repositorios.

3.7.1 Administrar Privilegios. -

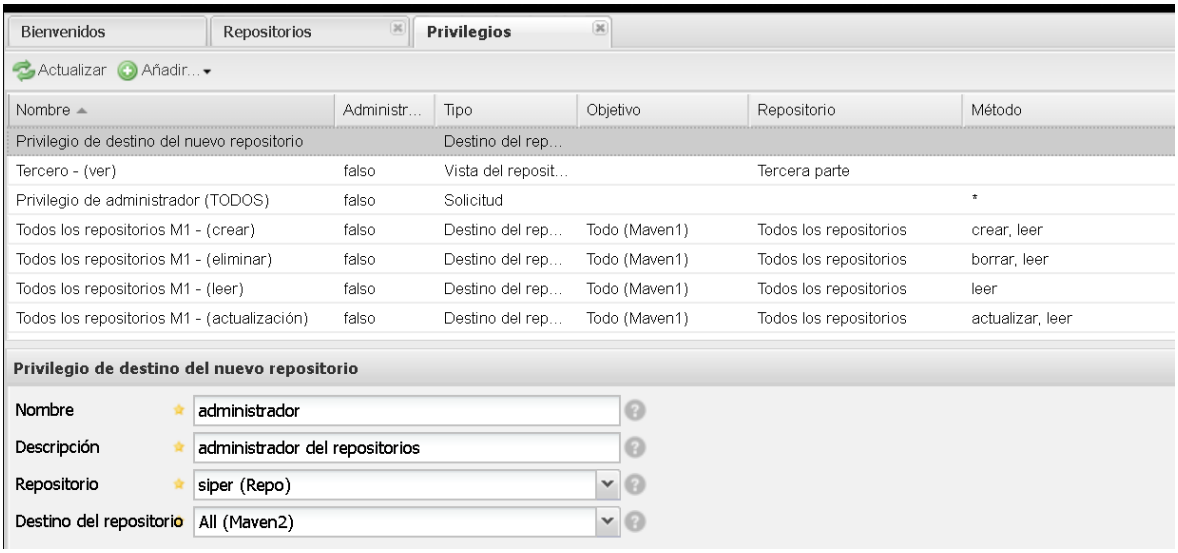
Puede acceder a la configuración de privilegios a través del elemento de menú Privilegios en el submenú Seguridad en el menú principal de la izquierda. El administrador del repositorio tiene tres tipos de privilegios: (Sonatype, 2008).

- Privilegios de la aplicación: abarca las acciones que un usuario puede ejecutar en la interfaz de usuario.
- Privilegios de destino del repositorio: regula el nivel de acceso que tiene un usuario a un repositorio o destino de repositorio en particular.
- Privilegios de visualización del repositorio: controla si un usuario puede ver un repositorio.

Detrás de las escenas, un privilegio está relacionada con una sola operación REST y método como crear, actualizar, eliminar y leer.

Figura 98:

Menú de privilegios



Nota: La imagen muestra la pestaña de privilegios. Elaborada por autor.

Para crear un nuevo privilegio, vaya al panel de seguridad de clin en privilegios luego haga clic en el botón Agregar y luego seleccione Privilegio de destino del repositorio**.**

Figura 109:

Creación de un nuevo privilegio

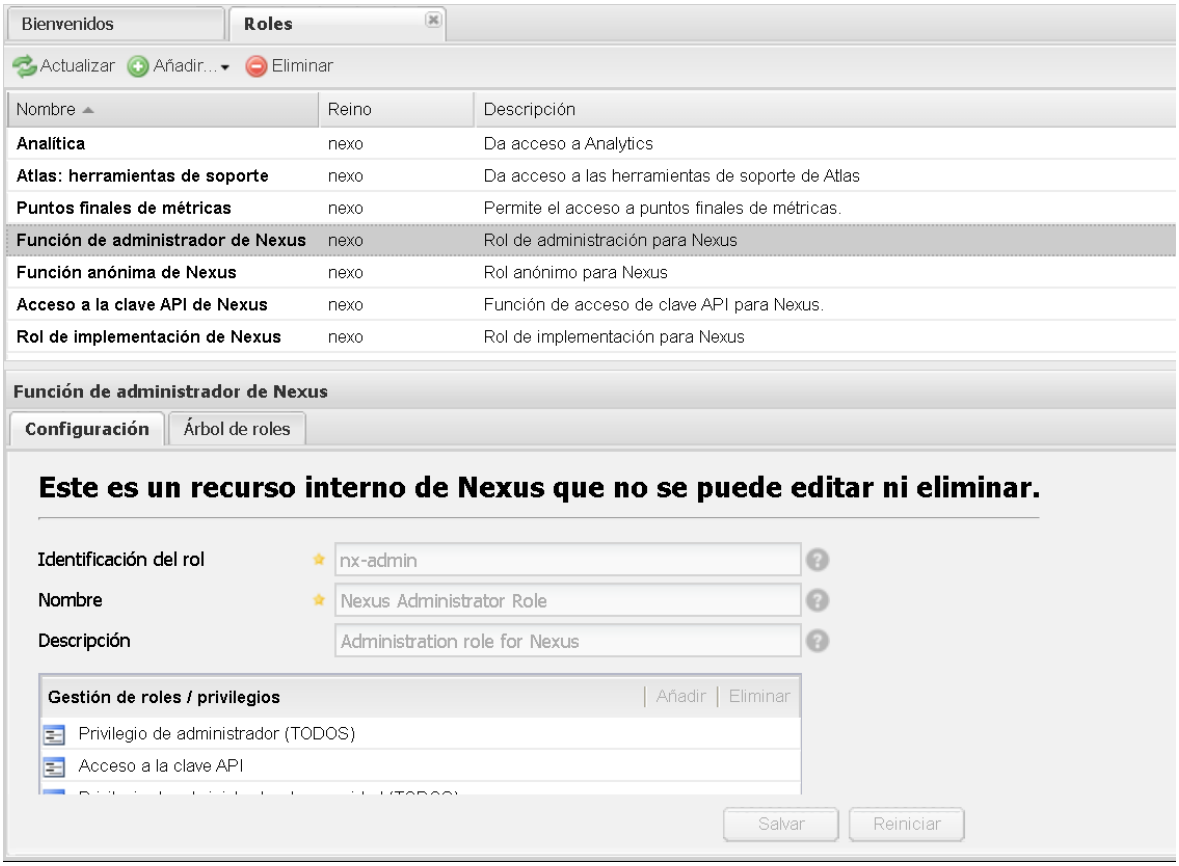
Nota: La imagen muestra la creación de un nuevo privilegio. Elaborada por autor.

3.7.2 Administrar Roles

El Administrador de Repositorios de Nexus incluye un gran número de funciones predefinidas incluyendo la administración de roles. Para poder configurar haga clic en el elemento del menú Roles bajo Seguridad en el menú principal para mostrar la lista de roles que se pueden configurar. (Sonatype, 2008).

Figura 20:

Administración de roles en Nexus



Nota: La imagen muestra la forma como crear un nuevo rol. Elaborada por autor.

Para crear un nuevo rol, debe hacer clic en el botón de añadir y selecciones Nexus.

Figura 211:

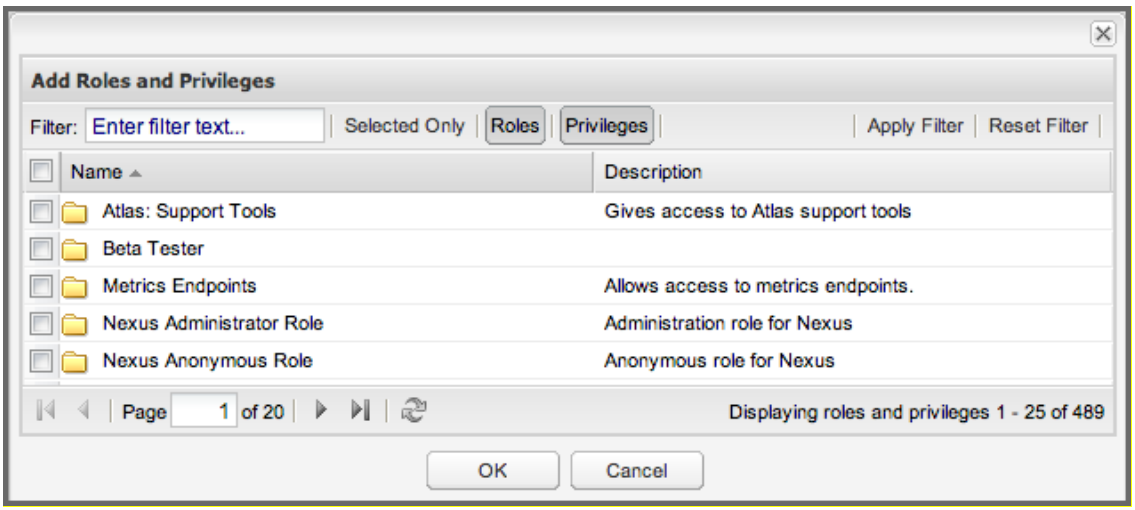
Creación de un nuevo rol

Nota: En la imagen muestra la forma de cómo crear un nuevo rol. Elaborada por autor.

Al crear un nuevo rol, deberá proporcionar un ID de rol, un Nombre y una Descripción. Los roles se componen de otros roles y privilegios individuales. Para asignar un rol o privilegio a un rol, haga clic en el botón Agregar debajo de Administración de roles / privilegios para acceder al cuadro de diálogo Agregar roles y privilegios que se muestra en la Figura 6.33, “El diálogo para agregar roles y privilegios”. Le permite filtrar la visualización paginada de todos los roles y privilegios disponibles con un texto de filtro, así como limitar la búsqueda solo a roles o privilegios. Usando el filtro y la paginación, podrá encontrar rápidamente el rol o privilegio deseado. (Sonatype, 2008).

Figura 22:

Descripción de roles en general de los roles.



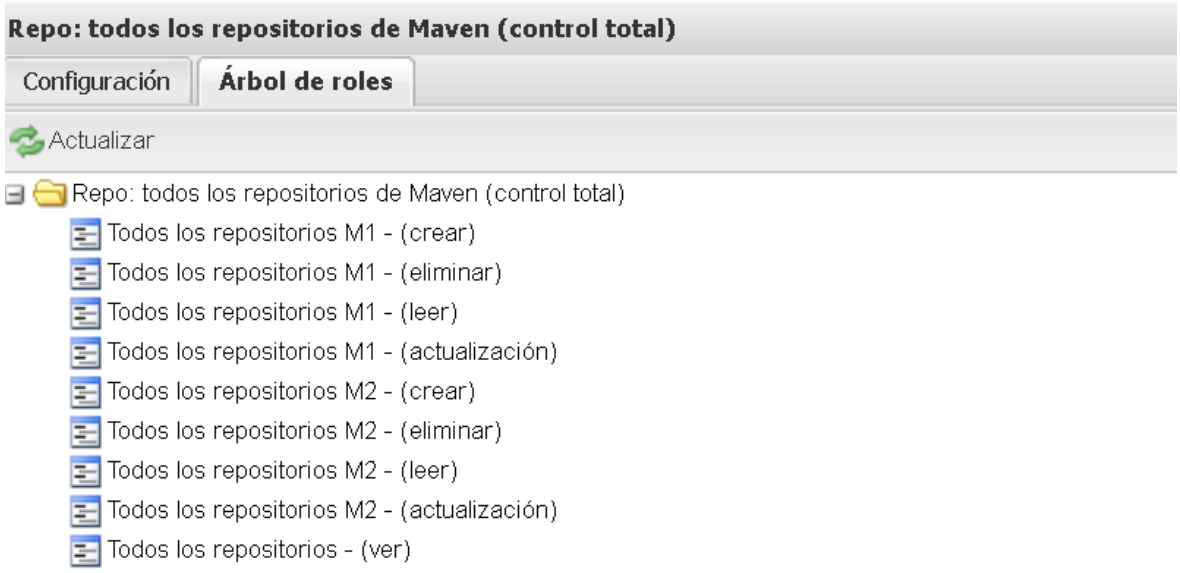
Nota: En la imagen muestra la descripción general de los roles. Elaborada por autor.

Nexus administra las funciones integradas y no se pueden editar ni eliminar. La sección de confirmación de roles debajo de la lista está visible pero deshabilitada para estos roles.

Un rol se compone de otros roles y privilegios individuales. Para ver los componentes de una función, seleccione la función en la lista de funciones y luego elija la pestaña Árbol de funciones.

Figura 23:

Árbol de funciones en la pestaña roles.



Nota: En la imagen muestra el árbol de funciones en la pestaña roles. Elaborada por autor.

3.7.3 Administrar Usuarios.

El Administrador de Repositorios de Nexus viene predefinido con tres usuarios: administrador**,** anónimo e implementación. El usuario administrador tiene todos los privilegios, el usuario anónimo tiene privilegios de solo lectura y el usuario de implementación puede leer e implementar en repositorios. (Sonatype, 2008).

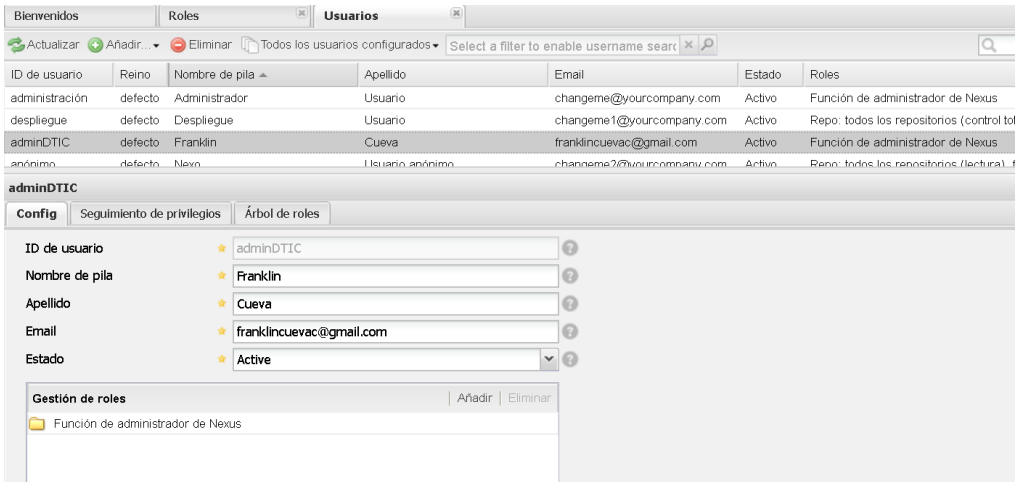
Para crear un nuevo usuario diríjase al panel lateral izquierdo en la pestaña seguridad haga clic en usuario en donde deberá llenar los campos como nombre, apellido y correo electrónico para poder crear el nuevo usuario.

Si necesita crear usuarios con un conjunto de permisos más específico, puede hacer clic en Usuarios en Seguridad en el menú principal de la izquierda. Una vez que vea la lista de usuarios, puede hacer clic en un usuario para editar nombre apellido y correo electrónico de ese usuario específico. Editar el estado de un usuario le permite

activar o desactivar un usuario por completo. También puede asignar o revocar roles específicos para un usuario en particular. (Sonatype, 2008).

Figura 24:

Creación de un nuevo usuario

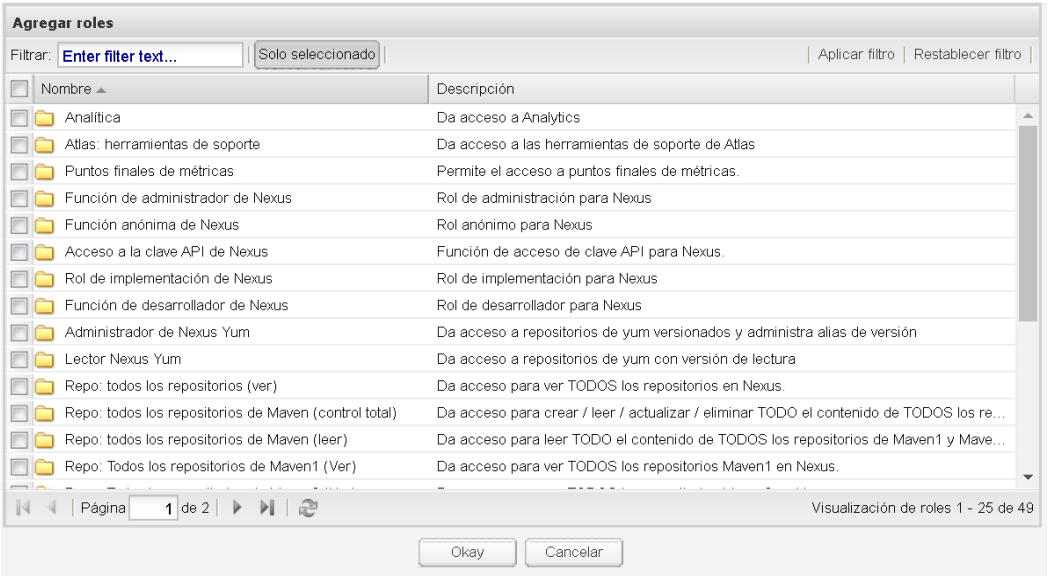


Nota: La imagen muestra la creación de un nuevo rol. Elaborada por autor.

Al hacer clic en el botón Agregar en la sección Administración de roles, aparecerá la lista de roles disponibles en una ventana emergente visible.

Figura 25:

Asignación de roles a un nuevo usuario

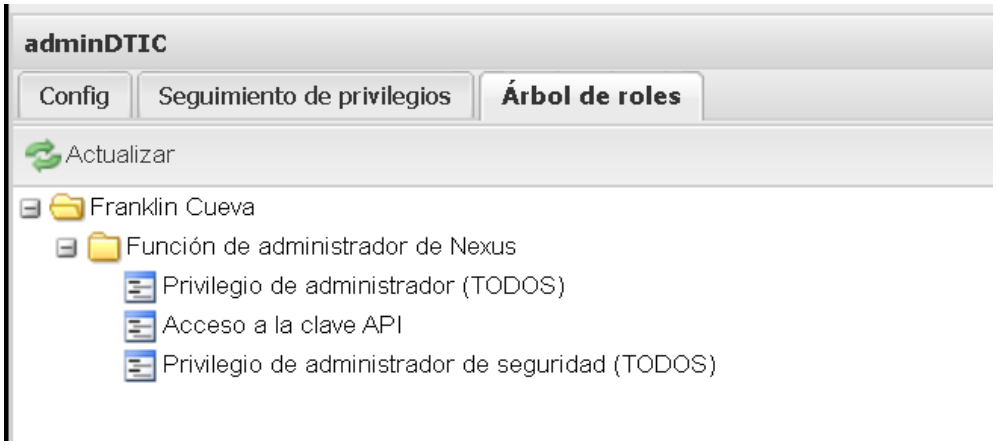


Nota: La imagen muestra la asignación de roles a un nuevo usuario. Elaborada por autor.

A un usuario se le pueden asignar uno o más roles que a su vez pueden incluir referencias a otros roles o privilegios individuales. Para ver seleccione el árbol de roles para un usuario en particular. (Sonatype, 2008).

Figura 26:

Árbol de roles de un Usuario.



Nota: En la imagen se muestra el árbol de roles de un Usuario.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones.

- LA Dirección de Tecnologías de la Información y Comunicaciones del Ejército Ecuatoriano es la unidad encargada del desarrollo de proyectos de software los mismo que usan metodologías y herramientas no adecuados para el desarrollo de los diferente aplicativos de la FF AA.
- Mediante la instalación del administrador de repositorios de Nexus ayudara a tener una administración centralizada de los distintos artefactos de software que ahí se producen, poder reutilizar el código fuente y ganar múltiples ventajas a la hora de programar nuevos aplicativos.
- Tiene múltiples ventajas la instalación del administrador de repositorios de Nexus ya que no dependemos de algo externo a la organización y puede mantener el control de las librerías propias producidas por el equipo de desarrolladores ya que son de carácter reservado y confidencia.

4.2 Recomendaciones.

- La DTIC debería actualizar sus diferentes herramientas utilizadas en el proceso de desarrollo de software para mejorar los tiempos de producción y la calidad del software.
- Capacitar al equipo de desarrolladores sobre el uso del Administrador de Repositorio de Nexus para poder poner en práctica y mejorar las condiciones de desarrollo de software.
- Delegar a una persona que sea el encargado de la administración de esta herramienta para poder llevar a cabo los diferentes procesos que demanda la administración del repositorio de Nexus y de esta manera aprovechar las

bondades que nos ofrece la misma.

- Aprovechar las diversas y múltiples ventajas que hoy en día nos ofrece el avance de la tecnología para mejores los procesos de fabricación de desarrollo de aplicativo web.
- Realizar el desarrollo de software utilizando el administrador de repositorios de Nexus para facilitar la reutilización de los artefactos de software.

REFERENCIAS BIBLIOGRÁFICAS

Andrango, M. E. (2017). *BENCHMARKING METODOLOGÍAS HÍBRIDAS PARA EL DESARROLLO DEL SOFTWARE PROTOTIPO SISTEMA DIDÁCTICO.* Ibarra: Universidad Técnica del Norte.

Besterfield, D. H. (2009). *Control de calidad.* México: Pearson Educación.

Bonilla , E., Díaz, B., Kleeberg, F., & Noriega, M. T. (2010). *Mejora Continua de los procesos.* Lima: Fondo Editorial.

Camisón , C., Cruz, S., & Gonzáles , T. (2006). *Gestión de la calidad.* Madrid: PEARSON EDUCACIÓN S.A.

Campderrich, F. B. (2013). *Ingeniería de Software.* Barcelona España: UOC.

Canos, J. H., Hetelier, P., & Penadés, C. (15 de 01 de 2006). *Metodologías Ágiles en el desarrollo de Software.* Obtenido de Metodologías Ágiles en el desarrollo de Software.: http://www.cyta.com.ar

Carro Paz, R., & Gonzáles Gómez, D. (2012). *Administración de la calidad total.* Mar del PLata: Facultad de Ciencias Económicas y Sociales.

Flower, M. (2006). *MartinFlower*. Obtenido de MartinFlower: https://martinfowler.com/

Fontalvo Herrera, T. J., & Vergara Schmalbach, J. C. (2010). *Gestión de la calidad en los Servicios ISO 9001:2008.* Eumed.net. Recuperado el 20 de Junio de 2020

Galvan, P. (s.f.). *SG Software Guru*. Obtenido de SG Software Guru: https://sg.com.mx/

GARZAS, J. (2014). *JAVIERGARZAS.COM*. Obtenido de JAVIERARZAS.COM: https://www.javiergarzas.com/2014/06/maven-en-10-min.html

Git. (s.f.). *Git*. Obtenido de Git: https://git-scm.com/

Gómez de León, F. C. (1998). *Tecnología del Mantenimiento Industrial.* Murcia: Servicio de Publicaciones, Universidad de Murcia.

Gutiérrez Pulido, H. (2010). *Calidad Total Y productividad.* México D.F: Mc Graw Hill.

ISO 25000. (2019). *ISO 25000*. Obtenido de ISO 25000: www.iso25000.com

ISO 9001. (2015). *ISO.* Recuperado el 10 de Junio de 2020, de www.iso.org

Izar Landeta, J. M., & González Ortiz, J. H. (2004). *7 Herramientas Básicas de la Calidad.* San Luis Potosi.

JAVA. (s.f.). *JAVA*. Obtenido de JAVA: https://www.java.com

León Avila, B. (11 de Diciembre de 2015). *Emprendices.* Recuperado el 15 de Mayo de 2020, de https://www.emprendices.co/la-importancia-la-calidad-las-empresas/

Lester, R. H., Enrick, N. L., & Mottley Jr., H. E. (1989). *Control de Calidad y Beneficio Empresarial.* Ediciones Díaz de Santos.

Logimov. (5 de Febrero de 2019). *Logimov.com*. Recuperado el 6 de Agosto de 2020, de https://www.logimov.com/blog/nwarticle/28/1/Que-es-un-kardex-para-control-de-inventario

Los Recursos Humanos.com. (Marzo de 2003). *Los Recursos Humanos.com*. Recuperado el 6 de Agosto de 2020, de Los Recursos Humanos.com: https://www.losrecursoshumanos.com/definición-de-alta-dirección/

Maida, E. G., & Pacienzia, J. (2015). Metodologías de Desarrollo de Software. En P. J. Maida EG, *Metodologías de Desarrollo de Software* (pág. 117). Argentina: Catedra Seminario de Sistemas.

Maldonado, J. A. (2015). *Fundamentos de la calidad total.*

Medium Corporation . (2020). *Medium Corporation* . Obtenido de Medium Corporation .

netbeans.apache.org. (2017). *netbeans.apache.org*. Obtenido de netbeans.apache.org: https://netbeans.apache.org/

NUEVA ISO 9001:2015. (11 de Junio de 2015). *NUEVA ISO 9001:2015*. Recuperado el 22 de Junio de 2020, de https://www.nueva-iso-9001-2015.com/2019/06/el-manual-de-calidad-funciones-uso-y-beneficios-de-su-empleo/

OpenWebinars. (s.f.). *OpenWebinars*. Obtenido de OpenWebinars: https://openwebinars.net

PowerData. (08 de 03 de 2008). *PowerData*. Obtenido de PowerData: www.powerdata.es

Pressman, R. S. (2010). *Ingeniería de Software. Un enfoque práctico.* Mèxico D.F: MC GRAW WILL.

Proceso de Desarrollo de Software. (01 de julio de 2012). *Proceso de Desarrollo de Software*. Obtenido de Proceso de Desarrollo de Software: http://ingsibran11iuty.blogspot.com/

Red Hat Customer Portal . (2020). *Red Hat Customer Portal* . Obtenido de Red Hat Customer Portal : https://access.redhat.com

Sonatype. (2008). *sonatype*. Obtenido de sonatype: https://www.sonatype.com

TutotialsPoint . (2019). *TutorialsPoint*. Obtenido de TutorialsPoint: www.tutorialspoint.com

Yánez, C. (05 de Diciembre de 2008). *Sistema de gestión de calidad en base a la norma ISO 9001.* Obtenido de Internacional Eventos: http://internacionaleventos.com/articulosISO.pdf

ANEXOS