Informe Técnico de Plan de Trabajo para Construcción de Software

Brayan Bladimir Castillo Figueredo

Estudiante

Marcia Elizabeth Solano Alvares

Instructora

Servicio Nacional de Aprendizaje (SENA)

Centro de Industria y Construcción

Análisis y Desarrollo de Software

2024

Introducción

Los sistemas de versionamiento o sistema de control de versiones, permiten a los usuarios realizar cambios en los proyectos de desarrollo de software y trabajar con otras personas de forma colaborativa, el trabajo que va a desempeñar cada desarrollador y los cambios de código que va a realizar se pueden distribuir en diferentes espacios de trabajo o ramas, por lo que los cambios que realice un desarrollador no afectarán al otro. Estas ramas se podrán unificar con otras ramas y cada cambio realizado quedará guardado en caso de que sea necesario volver a esa versión del código. El software de control de versiones guarda los cambios en un repositorio, por lo que, si los desarrolladores cometen un error, este se puede deshacer. Las tecnologías de versionamiento más conocidas son: Git, Subversión (SVN), Mercurial, Perforce, CVS(Concurrent Version System), Bitbucket, Team Foundation Server (TFS), AWS CodeCommit, etc.

A continuación, se hablará estas tecnologías de versionamiento y cual se va a usar para el desarrollo del proyecto.

Objetivos

Objetivo General

Elegir la herramienta de versionamiento que esté más acorde con el proyecto de desarrollo que se está elaborando.

Objetivos Específicos

- Investigar cuales son las tecnologías de versionamiento más usadas.
- Conocer las características de cada una de las tecnologías de versionamiento más usadas nivel mundial.
- Comparar las tecnologías de versionamiento para seleccionar la más idónea.

Tecnologías de Versionamiento

1. Git

Según (Atlassian, s.f.) Git es un sistema de control de versiones, es el más usado a nivel mundial es un proyecto de código abierto desarrollado por Linus Torvalds. Git tiene una arquitectura distribuida, es un ejemplo de DVCS (Sistema de Control de Versiones Distribuido). Además de tener una arquitectura distribuida, Git se ha diseñado teniendo en cuenta el rendimiento, seguridad y flexibilidad.

- **Rendimiento:**

  La confirmación de nuevos cambios, la ramificación y confirmación de versiones anteriores se optimizan en favor del rendimiento. Git se centra en el contenido del archivo, cuyo formato de objeto de los archivos del repositorio de Git emplean una combinación de codificación delta (que almacena diferencias de contenido) y compresión, y guarda el contenido de directorios y objetos de metadatos de las versiones.

- **Seguridad:**

  Git tiene como principal prioridad la integridad del código fuente gestionado, contenido de archivos y verdaderas relaciones entre estos y los directorios, las versiones, etiquetas y confirmaciones, todos objetos del repositorio de Git, todos protegidos con un algoritmo de hash llamado SHA1. Con esto se puede tener la certeza de contar con un auténtico historial de contenido del código fuente.

- **Flexibilidad:**

  Git es flexible en la capacidad para varios tipos de flujos de trabajo desarrollado no lineal, su eficiencia en proyectos tanto grandes como pequeños y su compatibilidad

con numerosos sistemas y protocolos. Posibilita la ramificación y etiquetado como procesos de primera importancia y las operaciones que afectan a las ramas y etiquetas, también almacenándose en el historial de cambios, no todos los sistemas de versiones ofrecen este nivel de seguimiento.

2. Subversion (SVN)

Según (Collins-Sussman, W. Fitzpatrick, & Pilato, 2004) es un sistema de control de versiones libre y de código fuente abierto, manejando ficheros y directorios a través del tiempo, en donde existe un árbol de ficheros en un repositorio central, este servidor es como un repositorio de ficheros ordinario, recordando todos los cambios hechos a sus ficheros y directorios, permitiendo examinar o recuperar versiones antiguas. La posibilidad de acceder a un repositorio de red desde distintos ordenadores fomenta la colaboración del equipo. Subversion es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros.

Subversion proporciona:

- Versionado de directorios.
- Verdadero historial de versiones.
- Envíos atómicos.
- Versionado de metadatos.
- Elección de las capas de red.
- Manipulación consistente de datos.
- Ramificación y etiquetado eficientes.

- Está implementado como una colección de bibliotecas compartidas en C con APIs bien definidas.

3. Mercurial

Es un sistema de control de versiones distribuido, libre y gratuito. Es usado para gestionar proyectos de gran tamaño, siendo compatible con sistemas operativos como Microsoft Windows, Linux, FreeBSD y macOS. Esta tecnología fue lanzada en el año 2005 por Matt Mackall, destacando por su facilidad de uso y simplicidad, convirtiéndolo en una opción atractiva para desarrolladores y equipos que buscan una herramienta que sea poderosa pero intuitiva. Esta tecnología de caracteriza por:

- Ser multiplataforma.
- Estar implementado principalmente en Python, pero también en C y Rust.
- Tener un modelo basado en conjuntos de cambios.
- Tener un sistema de ramificación simple y explícito.
- Tener comandos intuitivos y terminología accesible.

4. Perforce

Es un sistema de control de versiones comercial, utilizado para gestionar el código fuente fuente de proyectos, permitiendo a los equipos de desarrollo trabajar de forma eficiente y segura. Fue creado por Christopher Seiwald en 1995, y a estado evolucionando a lo largo de los años para convertirse en una suit completa de herramientas para el desarrollo de software, incluyendo no solo versiones, sino también gestión de proyectos, colaboración en equipo y automatización de procesos. Algunas de las características que tiene perforce son:

- Control de versiones centralizado almacenando todas la versiones en un servidor central.
- Puede manejar proyectos de cualquier tamaño.
- Ofrece robustas medidas de seguridad para proteger el código fuente y garantizar la integridad de datos.
- Incluye funcionalidades como ramificación, fusión, etiquetado, revisión de código y búsqueda avanzada.
- Ofrece diferentes opciones de licenciamiento para adaptarse a las necesidades de la organización.

5. CVS (Concurrent Version System)

Es un sistema de control de versiones utilizado para gestionar el código fuente de los proyectos, fue una de las primeras tecnologías de este tipo en ser popular y ha sido fundamental en la evolución de los sistemas modernos de control de versiones como Git. Fue creado a finales de los 90s por Dick Grune, fue diseñado principalmente para permitir a múltiples desarrolladores trabajar de forma simultánea en un mismo proyecto, manteniendo el historial de todos los cambios realizados en el código. Algunas de las características que tiene CVS son:

- Almacena las versiones de los archivos en un servidor central.
- Permite el rastreo de los cambios realizados en el código a lo largo del tiempo, facilitando identificar lo errores y la recuperación de versiones anteriores.
- Permite crear ramas del código para desarrollar nuevas funcionalidades de forma aislada.
- Permite realizar los cambios combinados en diferentes ramas.

- Permite asignar etiquetas a versiones específicas del código para facilitar la identificación y seguimiento.

6. Bitbucket

Es una plataforma de desarrollo colaborativo basada en la nube que utiliza en el sistema de control de versiones de Git. Es común en los equipos que trabajan en proyectos de código abierto que utilizan lenguajes como Python y Java. Fue creado por Atlassian en el 2008, es una herramienta esencial que facilita la colaboración en proyectos de código, la gestión de repositorios de Git y la integración con otras herramientas de desarrollo. Algunas de las características principales de Bitbucket son:

- Permite la creación de repositorios de Git privados y públicos, brindando flexibilidad para gestionar diferentes tipos de proyectos.
- Facilita la revisión de código por parte de otros miembros del equipo, ayudando a mantener la calidad del código e identificar posibles errores.
- Se integra de forma nativa con Jira, lo que permite vincular tareas y errores con commits de Git.
- Permite automatizar la construcción, prueba y despliegue de aplicaciones, agilizando el proceso de desarrollo.
- Ofrece funcionalidades avanzadas de Branching y Mergin, permitiendo trabajar en diferentes versiones de código de forma simultánea.
- Incluye una Wiki para documentar proyectos, lo que facilita la colaboración y comunicación entre miembros del equipo.
- Permite solicitar la fusión de cambios en el código principal, lo que promueve a revisión y discusión de cambios antes de ser incorporados.

Conclusión

Existen muchas tecnologías que ser usan para el versionamiento de código, lo que facilita el trabajo entre los diferentes miembros de un equipo y la posibilidad de recuperación de versiones de código anterior en caso de que existan errores. La herramienta más usada por las empresas en la actualidad es Git por su versatilidad, posibilidad, velocidad y capacidad para almacenar grandes cantidades de código.

Referencias

Atlassian. (s.f.). *atlassian.com*. Obtenido de Qué es Git: https://www.atlassian.com/es/git/tutorials/what-is-git

Collins-Sussman, B., W. Fitzpatrick, B., & Pilato, C. (2004). *svnbook.red-bean.com.* Obtenido de Control de versiones con Subversion: https://svnbook.red-bean.com/es/1.0/index.html