RAP 15_ GA7_AA1_EV01_IC_INFORME TÉCNICO DE PLAN DE TRABAJO PARA CONSTRUCCIÓN DE SOFTWARE

LUZ ANGELA GUTIÉRREZ GUTIÉRREZ

ESNEYDER ALEXANDER QUEVEDO GARCÍA

DIEGO ALEXANDER TOLOZA CASTRO

DANIEL VALENCIA SOTO

SERVICIO NACIONAL DE APRENDIZAJE SENA

2627044 PROGRAMA DE ANÁLISIS Y DESARROLLO DE SOFTWARE ADSO

ILMER CUELLO GARCIA

19 DE NOVIEMBRE DEL 2023

Tabla de Contenidos

Introducción

En este documento se elegirá como se manejará el versionameinto de la fase de codificación. Git es una herramienta invaluable para los desarrolladores pues permite trabajar en equipo sin correr el riesgo de que algún cambio en el software pueda afectar al trabajo realizado por los demás codificadores.

A la vez que Git permite manejar el versionamiento, hay herramientas basadas en Git que permiten usar el almacenamiento en la nube y facilitan el mantenimiento del código. Entre estas se hallan BitBucket, GitHub y GitLab.

Objetivo

Objetivo General

Aprender sobre Git y sus ventajas para el desarrollo de código.

Objetivos Específicos

Estudiar la forma de funcionamiento de Git.

Analizar las diferentes maneras de usar Git en la nube.

Definir que herramientas se usaran para el proceso de codificación en la fase 3 Ejecución.

Informe

¿Que es control de versiones?

El control de versiones, o "control de código fuente", es la práctica de rastrear y gestionar los cambios en el código de software. Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.

Los sistemas de control de versiones son herramientas de software que ayudan a los equipos de software a gestionar los cambios en el código fuente con el paso del tiempo y así es posible trabajar de forma más rápida e inteligente. Son especialmente útiles para los equipos de DevOps, ya que les ayudan a reducir el tiempo de desarrollo y a aumentar las implementaciones exitosas.

¿Como funciona?

El software de control de versiones realiza un seguimiento de todas las modificaciones en el código en un tipo especial de base de datos. En el caso de que se cometa un error, los desarrolladores pueden ir hacia atrás en el tiempo y comparar las versiones anteriores del código para ayudar a resolver el error, permitiendo minimizar las interrupciones para el equipo.

Este proceso puede hacerse de forma manual, pero no es muy eficiente, por eso se han desarrollado herramientas que faciliten esta gestión dando lugar a los llamados sistemas de control de versiones o VCS (del inglés Version Control System). Estos programas facilitan la administración de las distintas versiones de cada software, así como las posibles especializaciones realizadas. Algunos ejemplos de este tipo de herramientas son entre otros: CVS, Subversion, SourceSafe, ClearCase, Darcs, Bazaar, Plastic SCM, Git, SCCS, Mercurial, Perforce, Fossil SCM, Team Foundation Server.

A pesar de que el control de versiones se usa principalmente en la industria informática para controlar las distintas versiones del código fuente dando lugar a los sistemas de control de código fuente o SCM (siglas del inglés Source Code Management), estos mismos conceptos son aplicables a otros ámbitos como manejo de documentos, imágenes, sitios web, etc.

Características

Un sistema de control de versiones debe proporcionar:

- Forma de almacenar los elementos que deba gestionar (ej. archivos de texto, imágenes, documentación...).
- La posibilidad de realizar cambios sobre los elementos almacenados (ej. modificaciones parciales, añadir, borrar, renombrar o mover elementos).
- Mantener un registro histórico de las acciones realizadas con cada elemento o conjunto de elementos y también permitir volver a un estado anterior del producto.

También es útil el poder generar informes con los cambios entre dos versiones, informes de estado, marcado con nombre identificativo de la versión de un conjunto de ficheros, etc.

De todos los sistemas de control de versión existentes, los más usados son BitKeeper, Mercurial y Git, siendo este ultimo el mas popular en la actualidad.

¿Que es Git?

Git es un proyecto de código abierto con un mantenimiento activo que fue desarrollado originalmente por Linus Torvalds, quien también creo el kernel del sistema operativo Linux, en 2005.

Hoy en día, muchos proyectos de software dependen de Git para el control de versiones, incluidos proyectos comerciales y de código abierto. Git funciona muy bien en una amplia variedad de sistemas operativos e IDE (entornos de desarrollo integrados).

En Git, la copia de trabajo del código de cada desarrollador es también un repositorio que puede albergar el historial completo de todos los cambios lo que se llama DVCS (sistema de control de versiones distribuido, por sus siglas en inglés), a diferencia de los primeros sistemas de control de versiones que contaban con un único espacio para todo el historial de versiones del software, como CVS o Subversion (SVN).

Estos repositorios locales en el equipo de cada desarrollador permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y luego sincronizan su copia del repositorio con la del servidor. Esta forma de

trabajo es distinta del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones.

Aspectos básicos de Git

Cada vez que se guarda el trabajo, Git crea una confirmación. Una confirmación es una imagen de todos los archivos en un momento dado. Si un archivo no ha cambiado de una confirmación a la siguiente, Git usa el archivo anterior.

Las confirmaciones se vinculan a otras confirmaciones, creando un gráfico del historial de desarrollo. Es posible revertir el código a una confirmación anterior, inspeccionar cómo cambian los archivos de una confirmación a la siguiente y revisar información como dónde y cuándo se realizaron los cambios. Las confirmaciones se identifican en Git mediante un hash criptográfico único del contenido de la confirmación. Dado que todo tiene hash, es imposible realizar cambios, perder la información o dañar los archivos sin que Git lo detecte.

Cada desarrollador guarda los cambios en su propio repositorio de código local, llamado ramas (branches en inglés). Git proporciona herramientas para aislar los cambios y volver a combinarlos posteriormente. Las ramas administran esta separación. Una vez finalizado el trabajo creado en una rama, se puede combinar de nuevo en la rama principal (o troncal) del equipo.

Todos los archivos almacenados en Git se encuentran en uno de tres estados: modificados, almacenados provisionalmente o confirmados. Cuando se modifica un archivo por primera vez, los cambios solo existen en el directorio de trabajo en el equipo del desarrollador. El desarrollador debe almacenar provisionalmente los archivos modificados que se incluirán en la confirmación. El área de almacenamiento provisional (staged) contiene todos los cambios que se incluirán en la siguiente confirmación. Una vez que el desarrollador esté satisfecho con los archivos almacenados provisionalmente, los archivos se empaquetan como una confirmación con un mensaje que describe lo que ha cambiado (commit). Esta confirmación pasa a formar parte del historial de desarrollo.

Se recomienda dividir los cambios grandes en una serie de confirmaciones más pequeñas, así es más fácil revisar el historial de confirmaciones para buscar cambios de archivo específicos.

Ventajas de Git

Desarrollo simultáneo: Cada usuario tienen su propia copia local de código y pueden trabajar simultáneamente en sus propias ramas. Git funciona sin conexión, ya que casi todas las operaciones son locales.

Versiones de lanzamiento más rápidas: Las ramas permiten un desarrollo flexible y simultáneo. La rama principal contiene código estable y de alta calidad desde el que publica. Las ramas de cada desarrollador contienen trabajo en curso y se combinan con la rama principal tras la finalización. Al separar la rama de versión del desarrollo en curso, es más fácil administrar código estable y enviar actualizaciones más rápidamente.

Integración incorporada: Debido a su popularidad, Git esta integrado en la mayoría de las herramientas y productos. Todos los IDE más usados tienen compatibilidad integrada con Git y muchas otras herramientas admiten la integración y la implementación continuas, las pruebas automatizadas, el seguimiento de los elementos de trabajo, las métricas y la integración de características de informes con Git. Esta integración simplifica el flujo de trabajo diario.

Sólido soporte técnico de la comunidad: Git es de código abierto (open source) y se ha convertido en el estándar de facto para el control de versiones. El gran volumen de soporte técnico de la comunidad para Git en comparación con otros sistemas de control de versiones facilita recibir ayuda.

Git funciona con cualquier equipo: Es posible instalarlo en sistemas Windows, Linux, Mac y otros.

Git en la nube

Existen en el mercado diferentes herramientas para el manejo del control de versiones usando Git en la nube. Las mas comunes son BitBucket, GitLab y GitHub.

Nosotros hemos escogido trabajar con GitHub dado que es la plataforma más usada por los desarrolladores a nivel global.

Conclusiones

El uso de sistemas de control de versionamiento es una herramienta muy útil para los desarrolladores pues permite controlar el proceso de escritura de software, garantizando la integración entre los miembros del equipo y su trabajo en una única rama, permitiendo volver atrás en caso de errores.

Aunque Git puede ser usado de manera local, su mayor flexibilidad y potencia se evidencia cuando de usa entre un equipo de desarrolladores en diferentes lugares del mundo usando la nube como el repositorio de todo el código.

Eligiendo Git para nuestro desarrollo, con el apoyo de GitHub como sistema de almacenamiento en la nube, nos garantiza la mejor experiencia de trabajo, dado que todos lo miembros del equipo nos encontramos en diferentes partes del país.

Referencias

Sena. Material de aprendizaje “Integración Continua”

https://learn.microsoft.com/es-es/devops/develop/git/what-is-git

https://www.atlassian.com/es/git/tutorials/what-is-git

https://es.wikipedia.org/wiki/Git

https://es.wikipedia.org/wiki/Control_de_versiones