Programación III

Kelyn Tejada

Tarea 3

Luis Angel De La Cruz

2021-1031

25/11/2023

¿Qué es git?

Git es un sistema de control de versiones distribuido, lo que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan la copia del repositorio con la del servidor. Este paradigma es distinto del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones.

La flexibilidad y popularidad de Git lo convierten en una excelente opción para cualquier equipo. Muchos desarrolladores y graduados universitarios ya saben cómo usar Git. La comunidad de usuarios de Git ha creado recursos para entrenar a los desarrolladores y la popularidad de Git facilita recibir ayuda cuando se necesita. Casi todos los entornos de desarrollo tienen compatibilidad con Git y las herramientas de línea de comandos de Git implementadas en todos los sistemas operativos principales.

¿Para que funciona git init?

git init es un comando de Git fundamental para iniciar un repositorio nuevo en un directorio específico. Al ejecutarlo, Git establece la estructura básica de control de versiones en ese directorio creando el directorio. git, donde se almacenan los metadatos y la base de datos de objetos del repositorio. Esto permite empezar a rastrear cambios en los archivos presentes en ese directorio y marca el punto inicial para comenzar a realizar commits y mantener un historial de versiones del proyecto. Es el primer paso para utilizar Git en un proyecto, habilitando el seguimiento de cambios y la gestión de versiones de forma eficiente.

¿Qué es una rama?

Una rama en Git es una línea de desarrollo independiente que permite a los equipos de desarrollo trabajar en paralelo sin afectar el código en otras ramas. Imagina el proyecto como un árbol: la rama principal (o rama "master" en muchos casos) es la línea principal de desarrollo.

Cuando creas una nueva rama, estás divergiendo de esa línea principal para trabajar en funcionalidades, correcciones de errores o cualquier modificación sin alterar directamente la rama principal.

Las ramas permiten a los desarrolladores trabajar de forma aislada en sus tareas y luego fusionar esos cambios de regreso a la rama principal o a otras ramas, integrando así el trabajo de diferentes colaboradores. Esto facilita la colaboración, el desarrollo de características nuevas y la corrección de errores sin interferir directamente con la versión principal del proyecto hasta que se esté listo para fusionar los cambios.

¿Como saber en que rama estoy?

Puedes verificar en qué rama te encuentras actualmente utilizando el comando git branch. Este comando te mostrará todas las ramas presentes en tu repositorio local y resaltará con un asterisco (*) la rama en la que te encuentras.

Para obtener información más detallada sobre tu estado actual y la rama en uso, también puedes utilizar git status. Este comando no solo te muestra el estado de los archivos (archivos modificados, agregados, etc.), sino que también indica la rama actual y si tienes cambios pendientes que aún no se han agregado para el siguiente commit.

¿Quién creo git?

Git es un proyecto de código abierto maduro y con un mantenimiento activo que desarrolló originalmente Linus Torvalds, el famoso creador del kernel del sistema operativo Linux, en 2005.

¿Cuáles son los comandos más esenciales de Git?

Los comandos más esenciales de Git te permiten realizar diversas tareas desde la línea de comandos para gestionar un repositorio. Aquí tienes algunos de los más fundamentales:

git init: Inicia un nuevo repositorio Git en un directorio.

git clone: Clona un repositorio Git existente desde una ubicación remota (como GitHub) a tu máquina local.

git add: Agrega cambios al área de preparación para el próximo commit. Puedes agregar archivos específicos (git add nombre_archivo) o todos los cambios (git add .).

git commit: Registra los cambios en el repositorio. Puedes incluir un mensaje descriptivo para el commit utilizando la bandera -m: git commit -m "Mensaje del commit".

git status: Muestra el estado actual de tu repositorio, incluyendo archivos modificados, agregados o sin seguimiento, y la rama en la que te encuentras.

git pull: Obtiene cambios desde un repositorio remoto y los fusiona con tu rama local actual.

git push: Sube cambios locales a un repositorio remoto, actualizando así la rama remota con tus cambios locales.

git branch: Lista las ramas en tu repositorio local. Con la bandera -a muestra también las ramas remotas.

git checkout: Cambia entre ramas o versiones específicas de archivos. Puedes cambiar a una rama existente (git checkout nombre_rama) o crear una nueva rama (git checkout -b nombre_nueva_rama).

git merge: Fusiona cambios de una rama a otra. Por ejemplo, puedes fusionar una rama de características en la rama principal (git merge nombre_rama_caracteristicas).

¿Qué es Git Flow?

GitFlow se define como un sistema de branching o ramificación o modelo de manejo de ramas en Git, en el que se usan las ramas principales y la feature. De modo que la rama feature la crean los desarrolladores para fusionarla con la rama principal, únicamente cuando cumpla con sus labores.

Este sistema se caracteriza, además, por incluir múltiples ramas de mayor duración y más commit o confirmaciones.

¿Que es trunk based development?

El Trunk Based Development (TBD) es una metodología de desarrollo de software que se enfoca en trabajar directamente sobre la rama principal del repositorio, evitando ramas largas y aisladas. Los desarrolladores realizan cambios pequeños y frecuentes en la rama principal, lo que promueve integraciones rápidas y continuas. Esto facilita entregas regulares de funcionalidades y mejoras, manteniendo la rama principal siempre lista para producción. Además, fomenta la colaboración directa y la revisión constante del código entre los desarrolladores, reduciendo así los conflictos de integración y mejorando la estabilidad general del proyecto.

Una característica clave del TBD es el enfoque en cambios pequeños y revisiones rápidas del código. Esto permite una retroalimentación más temprana, una resolución más ágil de problemas y la reducción del riesgo de conflictos de integración. Además, el uso de "features flags" o interruptores de características permite desarrollar funcionalidades de manera encapsulada, activándolas o desactivándolas de manera controlada. Esta estrategia facilita la implementación progresiva y segura de nuevas características en el software sin interrumpir su funcionamiento general.