Evidencia de Conocimiento: GA7-220501096-AA1-EV01 Informe Técnico de Plan de Trabajo Para Construcción de Software

Presentado por:

Johana Edith Melo Chaparro

Tecnología en Análisis y Desarrollo de Software

Presentado a: John Alexander Lamprea Hernández

Centro de la Tecnología del Diseño y la Productividad Empresarial

SENA (Servicio Nacional de Aprendizaje)

Girardot -Cundinamarca

Ficha N°: 2977481

Julio 06 de 2025

Contenido

Contenido de Imágenes

Introducción

Este trabajo tiene como objetivo explicar qué es Git, sus principales características, los estados que puede tener un archivo dentro de un proyecto manejado con Git, el flujo típico de trabajo con esta herramienta y una recopilación de los comandos más utilizados. A través de este recorrido, se busca proporcionar una comprensión clara y práctica del funcionamiento y la utilidad de Git en entornos de desarrollo profesional.

Objetivo

Analizar y comprender el funcionamiento del sistema de control de versiones Git, destacando su importancia en el desarrollo de software colaborativo

1. ¿Qué es el Sistema de Control de Versiones – Git?

Git es un sistema de control de versiones que nació en el año 2005 producto de las necesidades que afrontaba la comunidad del kernel de Linux de encontrar un mecanismo que les permitiera soportar las actualizaciones y mantenimientos que requería este gran proyecto. Según Git (2021), algunos de los objetivos trazados que debía cumplir este nuevo mecanismo incluían:

Ilustración 1 Objetivos Gif

Nota: imagen creada por Johana Melo

Git, a diferencia de otros sistemas de control de versiones, posee un sistema de almacenamiento de información en los que cada versión posee una copia de instantáneas en miniatura, es decir, ofrece la posibilidad de trabajo aun cuando no se tiene conectividad al servidor central, ya que localmente se posee toda la información requerida incluso si se desea hacer procesos de recuperación a versiones anteriores.

2. Características Clave de Git

- Distribuido y Descentralizado

A diferencia de los sistemas de control de versiones centralizados, en los que se almacena una única copia del repositorio, Git es distribuido y como desarrolladores tenemos una copia completa del historial de cambios. Esto nos permite trabajar de forma independiente y fusionar nuestros cambios en el repositorio principal de manera eficiente.

- Rastreo Preciso de Cambios

Git realiza un seguimiento preciso de los cambios realizados en cada archivo a lo largo del tiempo. Esto nos permite como desarrollador ver el historial completo de cambios, incluidas las diferencias entre versiones y la posibilidad de regresar a versiones anteriores en caso de problemas.

- Ramificación Eficiente

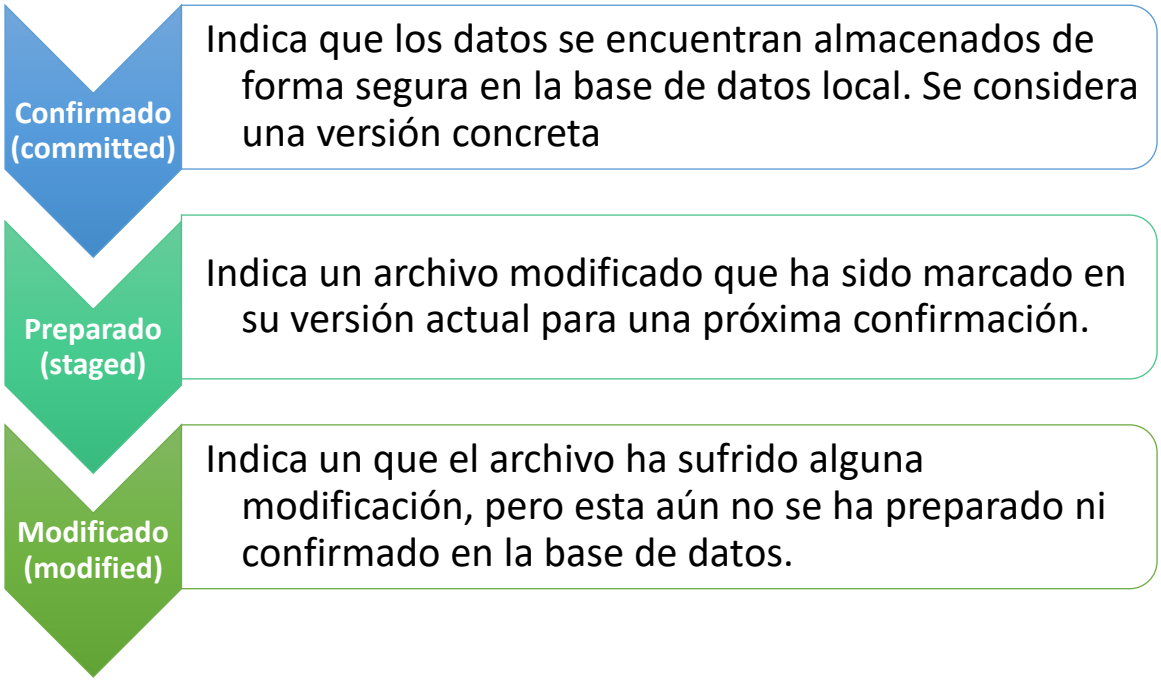
Git hace que la creación y gestión de ramas branches sea sencillo. Como desarrollador podemos crear ramas para trabajar con nuevas características o arreglos sin afectar la rama principal. Esto fomenta el desarrollo paralelo y facilita la colaboración en equipos grandes.

- Fusiones Simplificadas

La capacidad de fusionar cambios entre diferentes ramas es una característica clave de Git. Como desarrollador podemos combinar cambios de manera eficiente, lo que facilita la integración continua y evita conflictos de código. (Kranio, 2023)

3. ¿Cuáles son los Estados Principales Para Cada Uno de los Archivos de un Proyecto que Maneja Git?

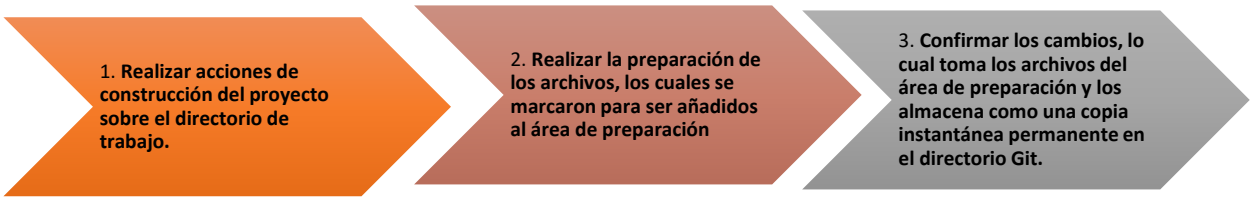
Ilustración 2 Estados principales Git



Nota: imagen creada por Johana Melo

4. ¿Cuál Sería un Flujo Normal de Acciones Para Trabajar con Git?

5. *Ilustración 3 Flujo normal de acciones*



Nota: imagen creada por Johana Melo

5. Comandos Básicos de Git

Algunos de los comandos más comunes que ayudan a manejar un repositorio GIT son:

git init: Crea un nuevo repositorio GIT en el directorio actual o en un nuevo proyecto.

git clone: Copia un repositorio existente, ya sea remoto o local, en tu máquina local.

git add: Agrega archivos al área de preparación (staging).

git commit: Guarda los cambios preparados en el repositorio con un mensaje descriptivo.

git status: Muestra el estado del repositorio, indicando qué archivos han sido modificados, añadidos o están listos para el commit.

git push: Envía tus commits locales al repositorio remoto.

git pull: Descarga y fusiona los cambios del repositorio remoto al local.

git branch: Muestra las ramas actuales, crea una nueva rama o elimina una existente.

git checkout: Cambia entre ramas o crea una nueva rama y cambia a ella.

git merge: Fusiona una rama con otra. Esto suele utilizarse después de terminar el desarrollo en una rama de características.

git log: Muestra el historial de commits del repositorio.

git reset: Deshace cambios locales y reinicia el área de preparación al último commit.

git rm: Elimina archivos del área de preparación y del directorio de trabajo.

git stash: Guarda temporalmente los cambios no confirmados y limpia el área de trabajo para poder cambiar de ramas o realizar otras tareas sin perder tu trabajo actual.

git fetch: Descarga los cambios del repositorio remoto sin fusionarlos inmediatamente. (Team, 2025)

Conclusiones

Conocer y dominar Git no solo mejora la productividad individual, sino que también fortalece la colaboración en proyectos grupales, haciendo que el proceso de desarrollo sea más seguro, ordenado y eficiente.

Git permite mantener un historial detallado de modificaciones, trabajar en diferentes ramas de desarrollo sin afectar la versión principal del proyecto y recuperar versiones anteriores en caso de errores.

Git es una herramienta fundamental en el desarrollo de software moderno gracias a su capacidad para gestionar eficientemente los cambios en el código y facilitar la colaboración entre múltiples desarrolladores

Bibliografía

Git. (2021) Git - Acerca del control de versiones. Git --Local-Branching-on-the-Cheap. https://Git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones

Kranio, T. (29 de 08 de 2023). *kranio*. Obtenido de kranio: https://www.kranio.io/blog/descubriendo-git-caracteristicas-y-ventajas

Team, S. (14 de 01 de 2025). *swhosting*. Obtenido de swhosting: https://www.swhosting.com/es/blog/comandos-basicos-de-git-guia-para-principiantes

Younes. (2021). Gitlab VS Github VS BitBucket. Which one deserve your time? DEV Community. https://dev.to/yafkari/gitlab-vs-github-vs-bitbucket-which-one-deserve-your-time-2npm