Nombre: Jonathan José

Apellido: Frías Martínez

Matricula: 2023-1117

Carrera: Desarrollo de software

Materia: Programación 3

Maestro: Kelyn Tejeda Belliard

Tema: Tarea 3 - Git

Tabla de contenido

¿Qué es Git?

Git es un sistema de control de versiones distribuido, creado para llevar un registro de los cambios realizados en archivos de un proyecto, especialmente en el desarrollo de software. Permite a varios desarrolladores trabajar de manera simultánea en el mismo código sin pisar el trabajo de los demás. Git guarda un historial completo de los cambios, permitiendo volver a versiones anteriores del proyecto en cualquier momento.

Una de las principales características de Git es que cada desarrollador posee una copia local completa del repositorio, lo que permite trabajar sin conexión y realizar cambios de forma independiente antes de sincronizarlos con el repositorio central.

¿Para qué sirve el comando git init?

El comando **git init** se utiliza para **inicializar un nuevo repositorio Git** en un directorio existente. Este comando crea una subcarpeta llamada. git, que contiene todos los archivos necesarios para el seguimiento de versiones del proyecto, incluyendo el historial de cambios, las configuraciones, y la estructura interna de Git.

Ejemplo:

git init

Después de ejecutarlo, el directorio comenzará a ser monitoreado por Git.

¿Qué es una rama en Git?

Una **rama (o branch)** en Git es una versión paralela del proyecto que permite desarrollar nuevas funcionalidades, corregir errores o experimentar con ideas, sin afectar el código principal (usualmente almacenado en la rama main o master).

Las ramas permiten que múltiples desarrolladores trabajen de forma independiente. Una vez que el trabajo en una rama está listo, se puede **fusionar (merge)** con la rama principal.

Ejemplo común de uso:

git checkout -b nueva-funcionalidad

Esto crea una nueva rama llamada nueva-funcionalidad y cambia a ella automáticamente.

¿Cómo saber en cuál rama estoy trabajando?

Para saber en qué rama estás trabajando actualmente, puedes usar:

git Branch

Este comando lista todas las ramas del repositorio y marca con un asterisco (*) la rama activa.

También puedes usar:

git status

Este comando te da información sobre el estado actual del repositorio, incluyendo la rama en la que estás trabajando.

¿Quién creó Git?

Git fue creado por **Linus Torvalds** en el año 2005. Torvalds es también conocido por haber iniciado el desarrollo del núcleo Linux. Git nació como una solución rápida y eficiente después de que el proyecto del núcleo de Linux dejara de utilizar otro sistema de control de versiones llamado BitKeeper, por motivos de licencias.

¿Cuáles son los comandos esenciales de Git?

Aquí tienes una lista con los comandos más comunes y esenciales de Git:

Comando	**Descripción**
git init	Inicializa un repositorio Git.
git clone [url]	Clona un repositorio remoto a local.
git status	Muestra el estado actual del repositorio.

git add [archivo]	Añade archivos al área de preparación (staging).
git commit -m "mensaje"	Guarda los cambios en el historial del proyecto.
git push	Envía los commits locales al repositorio remoto.
git pull	Trae los cambios del repositorio remoto y los fusiona.
git branch	Lista, crea o elimina ramas.
git checkout [rama]	Cambia de rama.
git merge [rama]	Fusiona una rama en la rama actual.

¿Qué es Git Flow?

Git Flow es un modelo de ramificación para organizar el trabajo en un proyecto Git de manera estructurada. Fue propuesto por Vincent Driessen en 2010 y define un flujo de trabajo basado en ramas que ayuda a manejar el desarrollo de software de manera ordenada.

Las ramas principales en Git Flow son:

- main (o master): contiene el código listo para producción.
- develop: contiene el código en desarrollo.
- feature/*: ramas para el desarrollo de nuevas funcionalidades.
- release/*: ramas para preparar versiones antes de producción.
- hotfix/*: ramas para corregir errores críticos en producción.

Este modelo es ideal para equipos con ciclos de desarrollo bien definidos y donde se requiere un control riguroso de versiones.

¿Qué es el desarrollo basado en trunk (Trunk Based Development)?

El **Trunk Based Development** es una estrategia de control de versiones en la que todos los desarrolladores trabajan directamente en la rama principal del proyecto (conocida como trunk, main o master). A diferencia de Git Flow, no se usan múltiples ramas para nuevas funcionalidades o correcciones.

En este modelo:

- Las ramas son cortas o incluso inexistentes.
- Los cambios se integran continuamente al trunk.
- Se utilizan prácticas como la integración continua (CI) y pruebas automatizadas para evitar errores.

Este enfoque es común en equipos que practican DevOps o metodologías ágiles como **Continuous Delivery**, ya que favorece la rapidez, simplicidad y colaboración continua.

Bibliografía:

1. Atlassian. (s. f.). *¿Qué es Git?* Atlassian. Recuperado el 2 de abril de 2025, de https://www.atlassian.com/es/git/tutorials/what-is-git
2. Git Documentation - https://git-scm.com/doc
3. Atlassian Git Tutorials - https://www.atlassian.com/git
4. Git Flow by Vincent Driessen - https://nvie.com/posts/a-successful-git-branching-model/
5. Atlassian. (s. f.). *git init*. Recuperado el 2 de abril de 2025, de https://www.atlassian.com/es/git/tutorials/setting-up-a-repository/git-init#:~:text=En%20comparaci%C3%B3n%20con%20SVN%2C%20el,proyectos%20con%20control%20de%20versiones.