Presentación

Nombre: *Abner Moisés Rabassa Javier*

Matricula: *2023-1829*

Asignatura: *Programación 3*

Profesor/a: *Kelyn Tejada Belliard*

Fecha: *1/4/2025*

Tarea 3

Tema: *Cuestionario*

Contents

1. ¿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.

2. ¿Para qué sirve el comando git init?

El comando git init crea un nuevo repositorio de Git. Puede utilizarse para convertir un proyecto existente y sin versión en un repositorio de Git, o para inicializar un nuevo repositorio vacío. La mayoría de los demás comandos de Git no se encuentran disponibles fuera de un repositorio inicializado, por lo que este suele ser el primer comando que se ejecuta en un proyecto nuevo.

Al ejecutar git init, se crea un subdirectorio de. Git en el directorio de trabajo actual, que contiene todos los metadatos de Git necesarios para el nuevo repositorio.

el comando git init permite crear de manera increíblemente sencilla nuevos proyectos con control de versiones. Con Git, no es necesario que crees un repositorio, importes los archivos ni extraigas una copia de trabajo.

3. ¿Qué es una rama en Git?

Una rama Git es simplemente un apuntador móvil apuntando a una de esas confirmaciones. La rama por defecto de Git es la rama master. Con la primera confirmación de cambios de realicemos, se creará esta rama principal master apuntando a dicha confirmación.

4. ¿Cómo saber en cuál rama estoy trabajando?

Para saber qué ramas están disponibles y cuál es el nombre de la rama actual, ejecuta git branch. El comando git branch se usa para gestionar y visualizar las ramas en un repositorio de Git. Dependiendo de cómo lo uses, puedes listar las ramas disponibles, crear nuevas ramas, cambiar su nombre o eliminarlas.

5. ¿Quién creó Git?

El desarrollo de Git fue iniciado por Linus Torvalds en abril de 2005 cuando el sistema propietario de control de versiones usado en el desarrollo del kernel de Linux desde 2002, BitKeeper revoco la licencia gratuita usada para el desarrollo del Linux. El titular del copyright de Bitkepper Larry McVoy, afirmó que Andrew Tridgell había creado Source Puller utilizando ingeniería inversa a partir de los protocolos de BitKeeper.

El diseño de Git se basó en BitKeeper y en Monotone. Originalmente fue diseñado como un motor de sistema de control de versiones de bajo nivel sobre el cual otros podrían codificar interfaces frontales, tales como Cogito o StGIT.

6. ¿Cuáles son los comandos esenciales de Git?

- Git clone
- Git branch
- Git checkout
- Git status
- Git add
- Git commit
- Git push
- Git pull
- Git revert
- Git merge

7. ¿Qué es Git Flow?

Gitflow es un modelo alternativo de ramificación de Git que implica el uso de ramas de características y múltiples ramas principales . Fue publicado y popularizado por primera vez por Vincent Driessen en nvie. En comparación con el desarrollo basado en troncos, Gitflow ofrece numerosas ramas de mayor duración y confirmaciones de mayor tamaño.

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

El desarrollo basado en trunk (Trunk-Based Development, TBD) es una estrategia de control de versiones en la que los desarrolladores trabajan en una única rama principal (trunk o main) y realizan integraciones frecuentes en ella en lugar de mantener múltiples ramas de larga duración. En este enfoque, las nuevas funcionalidades se desarrollan en ramas pequeñas y de corta duración o incluso directamente en el trunk, asegurando que los cambios se integren rápidamente y se validen mediante pruebas automáticas.

Bibliografía

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

https://www.atlassian.com/es/git/tutorials/setting-up-a-repository/git-init

https://git-scm.com/book/es/v2/Ramificaciones-en-Git-%C2%BFQu%C3%A9-es-una-rama%3F#:~:text=Una%20rama%20Git%20es%20simplemente,master%20apuntando%20a%20dicha%20confirmaci%C3%B3n.

https://www.atlassian.com/es/git/tutorials/using-branches/git-checkout#:~:text=Para%20saber%20qu%C3%A9%20ramas%20est%C3%A1n,rama%20actual%2C%20ejecuta%20git%20branch%20.&text=En%20el%20ejemplo%20anterior%2C%20se,este%20caso%2C%20la%20rama%20feature_inprogress_branch%20.

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

https://www.freecodecamp.org/espanol/news/10-comandos-de-git-que-todo-desarrollador-deberia-saber/

https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow#:~:text=Gitflow%20is%20an%20alternative%20Git,lived%20branches%20and%20larger%20commits.