TAREA 3.1 SISTEMA DE CONTROL DE VERSIONES (GIT)

IDS grupo 1P 2025A (Feb-Jun)

Michael Alexander Bernal Nápoles
25110114
Desarrollo de Software
I-1P
09/06/2025
Mtra. María Guadalupe Ortega Tirado
3er Parcial

Instrucciones.

Escucha y ve el video anexo y de ser necesario investiga en internet para contestar lo siguiente:

1. Escribe brevemente la historia de GIT
2. ¿Qué es GIT?
3. ¿Escribe a través de un diagrama cómo funciona GIT (Como se trabaja en GIT)?
4. ¿Qué es un repositorio?
5. ¿Por qué GIT es el VCS más usado y valorado en el mundo?
6. Escribe 5 ventajas del uso GIT

Punto 1: Historia de GIT.

Git fue creado en 2005 por Linus Torvalds (creador del núcleo de Linux). Todo comenzó como una respuesta a las limitaciones de otros sistemas de control de versiones centralizados. Git fue diseñado para ser eficiente y rápido, especialmente para manejar grandes proyectos como el kernel de Linux. Los siguientes eran los objetivos en mente a lograr con el nuevo sistema de control de versiones:

- Rápido
- Distribuido
- Seguro
- Adecuado para equipos de desarrollo grandes y descentralizados

Antes de Git, el equipo de Linux usaba un sistema llamado BitKeeper, pero su licencia era propietaria y eso generó conflictos. Finalmente, BitKeeper dejó de ser gratuito para la comunidad del núcleo de Linux, lo que llevó a Torvalds a desarrollar su propia herramienta: Git.

Torvalds desarrolló Git en unas pocas semanas para satisfacer esa necesidad, Git fue diseñado para ser distribuido, lo que significa que cada desarrollador tiene una copia completa del historial del proyecto en su máquina local.

Desde entonces, Git ha evolucionado enormemente y se ha convertido en el sistema de control de versiones más popular del mundo, usado por millones de desarrolladores y empresas (como Google, Microsoft, y muchas otras).

Punto 2: ¿Qué es GIT?

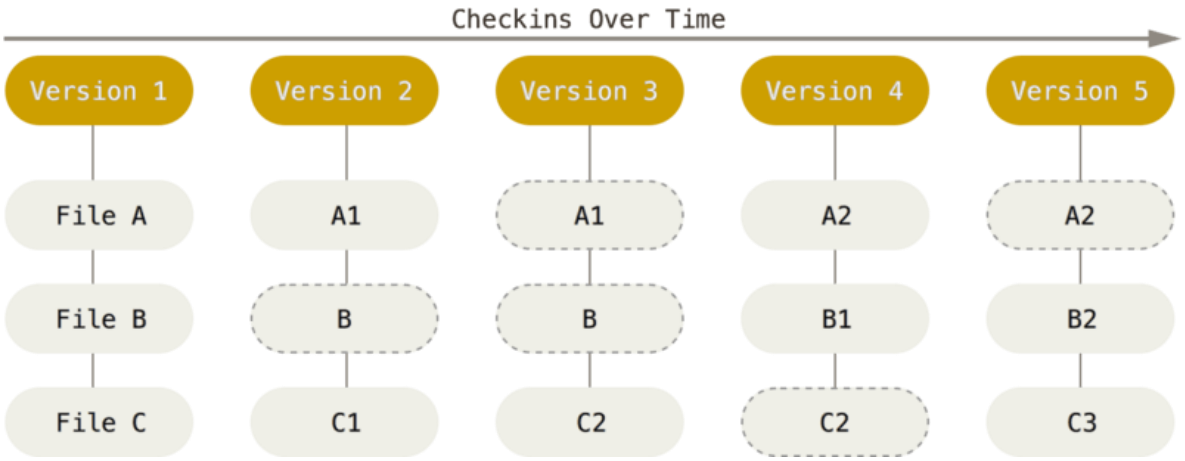
Git es un sistema de control de versiones. El control de versiones es un sistema que registra los cambios en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante. Git por su parte está distribuido diseñado para registrar, seguir y coordinar los cambios realizados en archivos (del código fuente) de manera rápida, eficiente y segura, permite a múltiples desarrolladores trabajar de forma simultánea en un mismo proyecto sin interferir entre sí, facilitando la colaboración y el seguimiento preciso del historial de cambios. Cada colaborador trabaja con una copia completa del repositorio, lo que ofrece independencia del servidor central y la posibilidad de trabajar sin conexión.

La principal diferencia entre Git y cualquier otro VCS (incluyendo Subversion y sus amigos) es la forma en la que manejan sus datos. Conceptualmente, la mayoría de los otros sistemas almacenan la información como una lista de cambios en los archivos. Estos sistemas (CVS, Subversion, Perforce, Bazaar, etc.) manejan la información que almacenan como un conjunto de archivos y las modificaciones hechas a cada uno de ellos a través del tiempo.

Para explicarse mejor, en el siguiente punto se abarca.

Punto 3: Explicar con un diagrama cómo funciona GIT (Como se trabaja en GIT).

Sobre el almacenamiento de datos en Git. Git **no** almacena los archivos como una secuencia de diferencias entre versiones (como hacen CVS o Subversion), sino que guarda una **instantánea completa** del contenido del proyecto **en cada commit**. Sin embargo, lo hace de forma eficiente**: si un archivo no ha cambiado** entre versiones, **Git no lo guarda de nuevo**, simplemente guarda una referencia al archivo ya almacenado.



Almacenamiento de datos como instantáneas del proyecto a través del tiempo.

Estructura interna de almacenamiento:

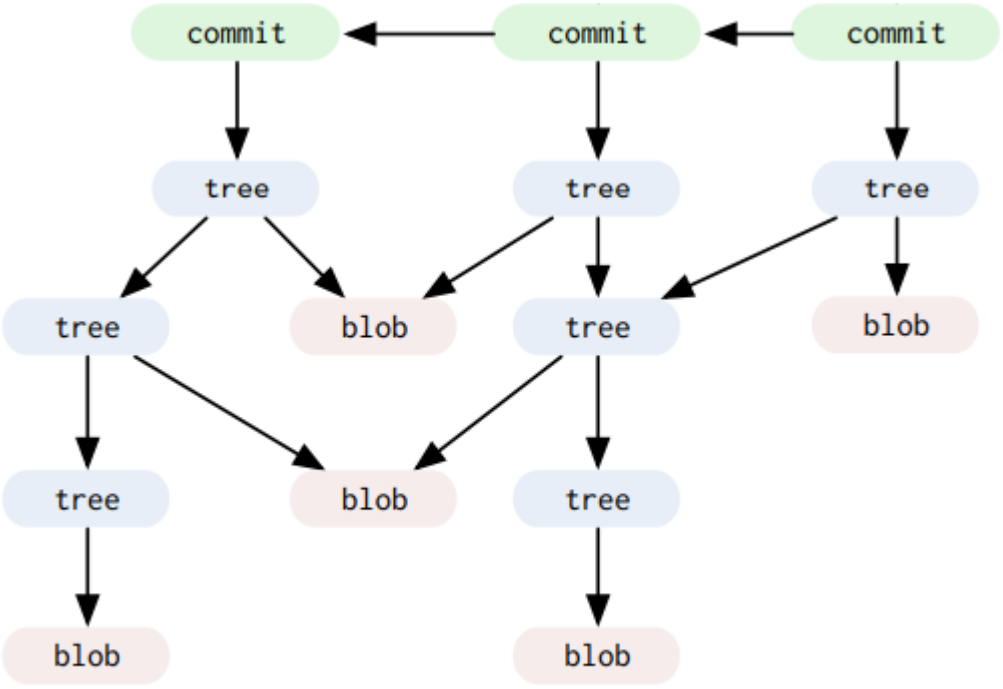
Git tiene una estructura basada en una base de datos de objetos. Hay cuatro tipos de objetos principales:

1. **Blob (Binary Large Object):** Representa el contenido de un archivo, Git guarda el contenido como un blob, y no se preocupa por el nombre del archivo ni su ubicación. Cada blob se identifica con un hash SHA-1 por lo que si dos archivos tienen el mismo contenido, comparten el mismo blob (ahorro de espacio gracias a la deduplicación).
2. **Tree (Árbol):** Representa un directorio, contiene referencias a blobs (archivos) y otros trees (subdirectorios), junto con sus nombres y permisos, esto es lo que le da estructura al contenido.
3. **Commit:** Representa un punto en el tiempo (una versión del proyecto).

    Contiene:

    - Un puntero a un objeto tree.
    - Información de autor y mensaje del commit y fecha.
    - Un puntero al commit padre (o varios en caso de un merge).

En resumen, git almacena los datos como una serie de instantáneas (snapshots) del proyecto. Cada vez que haces un commit, Git guarda una imagen completa del árbol de archivos, pero sólo los archivos que han cambiado se guardan como nuevos blobs. Todo se organiza a través de objetos con identificadores únicos basados en hashes criptográficos (SHA-1), lo que garantiza integridad y eficiencia.



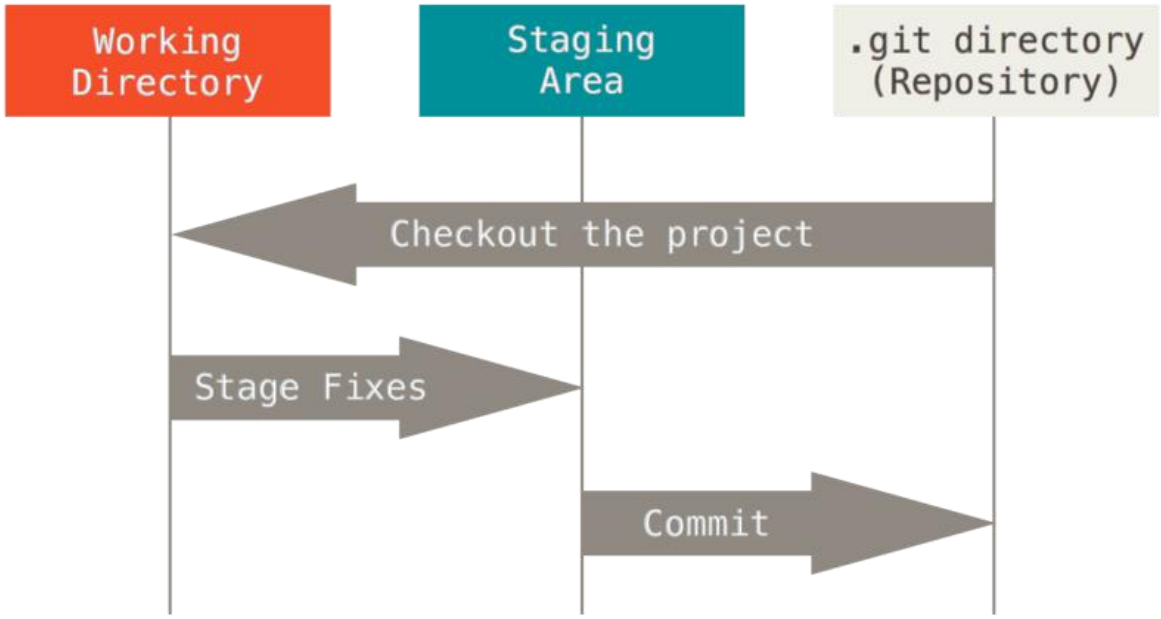
Punto 4: ¿Qué es un repositorio?

Un repositorio Git es el núcleo del control de versiones, es donde Git guarda todo el historial del proyecto, de forma distribuida, segura y eficiente. Puede estar en el equipo propio (repositorio local) o en un servidor (repositorio remoto) para trabajo colaborativo.

Git tiene tres estados principales en los que se pueden encontrar tus archivos: confirmado (committed), modificado (modified), y preparado (staged).

- Confirmado: significa que los datos están almacenados de manera segura en tu base de datos local.
- Modificado: significa que has modificado el archivo pero todavía no lo has confirmado a tu base de datos.
- Preparado: significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

Esto nos lleva a las tres secciones principales de un proyecto de Git: El directorio de Git (Git directory), el directorio de trabajo (working directory), y el área de preparación (staging area). (Ilustrado en la siguiente foto)



El directorio de Git es donde se almacenan los metadatos y la base de datos de objetos para tu proyecto. Es la **parte más importante de Git**, y es lo que se copia cuando clonas un repositorio desde otra computadora.

El directorio de trabajo es una copia de una versión del proyecto. Estos archivos se sacan de la base de datos comprimida en el directorio de Git, y se colocan en disco para que los puedas usar o modificar.

El área de preparación es un archivo, generalmente contenido en tu directorio de Git, que almacena información acerca de lo que va a ir en tu próxima confirmación. A veces se le denomina índice ("index"), pero se está convirtiendo en estándar el referirse a ella como el área de preparación.

El **flujo de trabajo básico** en Git es algo así:

1. Modificas una serie de archivos en tu directorio de trabajo.
2. Preparas los archivos, añadiéndolos a tu área de preparación.
3. Confirmas los cambios, lo que toma los archivos tal y como están en el área de preparación y almacena esa copia instantánea de manera permanente en tu directorio de Git.

Punto 5: ¿Por qué GIT es el VCS más usado y valorado en el mundo?

Por una gran variedad de razones entre ellas destacamos que Git es un sistema de control de versiones **distribuido**, lo que significa que cada desarrollador tiene una copia completa del historial del proyecto. Esto permite:

- Trabajar sin conexión.
- Recuperar versiones anteriores localmente.
- Mayor seguridad ante fallos del servidor central.

Ademas, Git permite crear, fusionar y eliminar ramas de forma eficiente. Esto fomenta flujos de trabajo como Git Flow, donde:

- Se desarrollan nuevas funcionalidades en ramas separadas.
- Se prueban antes de fusionarlas al proyecto principal.

Esta popularidad lo ha llevado a ser compatible con muchas plataformas populares como: GitHub, GitLab, Bitbucket, CI/CD (herramientas de revisión de código y despliegue automatizado) estas integraciones han potenciado su uso en entornos profesionales y colaborativos.

Finalmente, hay que mencionar que Git cuenta con una enorme comunidad de usuarios y desarrolladores lo que hace que exista abundante documentación y tutoriales, solución de problemas comúnmente documentados y mejora continua gracias al desarrollo abierto.

Punto 6: 5 Ventajas

1. Control de versiones distribuido

Cada desarrollador tiene una copia completa del repositorio (código, historial, ramas). Permite trabajar sin conexión y reduce el riesgo de pérdida de datos.

2. Velocidad y rendimiento

Operaciones como commit, merge o branch son muy rápidas. Git está optimizado para eficiencia, incluso en proyectos grandes.

3. Ramas ligeras y flexibles

Crear y gestionar ramas es rápido y seguro. Lo que facilita el trabajo en paralelo, pruebas de nuevas funciones y flujos como Git Flow.

4. Integración con herramientas modernas

Funciona perfectamente con plataformas como GitHub, GitLab y Bitbucket. Ademas soporta flujos de trabajo colaborativos, revisión de código y CI/CD.

5. Amplio soporte y comunidad

Git es software libre, con una enorme comunidad de usuarios, por lo que hay mucha documentación, soporte gratuito y mejora continua.

Bibliografía.

- Chacon, S., & Straub, B. (2014). *Pro Git* (2ª ed.). Apress. Recuperado de https://git-scm.com/book/en/v2
- Loeliger, J., & McCullough, M. (2012). *Version Control with Git: Powerful tools and techniques for collaborative software development* (2ª ed.). O'Reilly Media.
- Torvalds, L. (2005). *Git - Fast Version Control System*. Git.org (proyecto original de Git). Recuperado de https://github.com/git/git
- Preston-Werner, T. (2013). *The Git Parable*. Recuperado de https://tom.preston-werner.com/2009/05/19/the-git-parable.html