Estudiante

Karen Hidalgo

2022-0168

Maestro

Kelyn Tejada

Materia

Programación III

Desarrolla el siguiente Cuestionario

1-Que es Git?

Git es un sistema de control de versiones distribuido (DVCS, por sus siglas en inglés) diseñado para rastrear cambios en el código fuente durante el desarrollo de software. Fue creado por Linus Torvalds en 2005 y se ha convertido en una herramienta fundamental en el desarrollo colaborativo de software. Git se utiliza para gestionar y controlar el historial de cambios en archivos y carpetas en un proyecto, permitiendo a los desarrolladores colaborar de manera eficiente y mantener un registro completo de las modificaciones realizadas.

Git se utiliza comúnmente en combinación con servicios de alojamiento en la nube como GitHub, GitLab o Bitbucket, que proporcionan repositorios remotos para almacenar y colaborar en proyectos Git. Su popularidad se debe a su versatilidad, facilidad de uso y capacidad para gestionar proyectos complejos con equipos distribuidos.

2-Para que funciona el comando Git init?

El comando `git init` se utiliza para iniciar un nuevo repositorio de Git en un directorio. Cuando ejecutas este comando en un directorio que aún no es un repositorio de Git, Git configura todos los archivos y subdirectorios necesarios para comenzar a realizar un seguimiento de cambios en tu proyecto. Algunas funciones clave de `git init`:

Inicialización del Repositorio

- `git init` inicializa un nuevo repositorio de Git en el directorio actual.

Creación de Estructuras Internas

- Crea el subdirectorio `.git` que contiene todos los archivos necesarios para el control de versiones, como la configuración del repositorio, el historial de cambios y la información sobre las ramas.

Configuración Inicial

- Establece la configuración inicial del repositorio, incluyendo detalles como el nombre del autor y la dirección de correo electrónico. Estos detalles se utilizan en los registros de confirmaciones.

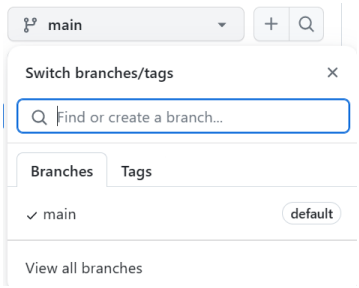
Listo para Realizar Seguimiento de Cambios

- Después de ejecutar `git init`, el directorio está listo para comenzar a realizar un seguimiento de cambios en archivos. Puedes comenzar a realizar confirmaciones (`git commit`), crear ramas (`git branch`), y trabajar con otros comandos de Git.

Punto de Inicio para Nuevos Proyectos

- `git init` es comúnmente utilizado cuando estás comenzando un nuevo proyecto y deseas utilizar Git para realizar un seguimiento de cambios en ese proyecto.

3-Que es una rama?



En Git, una rama (o branch en inglés) es una línea de desarrollo independiente que permite a los desarrolladores trabajar en características o problemas de forma aislada del código en otras ramas. Cada repositorio de Git comienza con una rama predeterminada llamada "master" (o "main" en versiones más recientes), y a partir de esta rama principal, puedes crear ramas adicionales para trabajar en nuevas funcionalidades, correcciones de errores u otros cambios sin afectar directamente la rama principal.

Características clave de las ramas en Git:

Aislamiento de Desarrollo

- Cada rama representa una línea independiente de desarrollo. Los cambios realizados en una rama no afectan directamente a otras ramas hasta que se fusionan.

Desarrollo Concurrente

- Varias ramas pueden existir simultáneamente, permitiendo que diferentes desarrolladores trabajen en diferentes características o problemas al mismo tiempo.

Facilita el Desarrollo por Funciones

- Es común crear una nueva rama para cada nueva característica o tarea. Esto facilita la gestión de cambios y la colaboración en equipos.

Histórico de Ramas

- Cada rama tiene su propio historial de cambios, lo que facilita el seguimiento de la evolución de cada característica o tarea.

Fusión de Ramas

- Puedes fusionar los cambios de una rama en otra cuando una característica está completa o un problema está resuelto. Git realiza la fusión de forma automática cuando es posible y, en caso de conflictos, solicita intervención manual.

Rama Principal (Master o Main)

- La rama principal es la línea principal de desarrollo. Es la rama a la que se fusionan otras ramas cuando las características están completas y se han probado.

5-Quien creo git?

Git fue creado por Linus Torvalds, el mismo individuo que también inició el desarrollo del kernel de Linux. Linus Torvalds desarrolló Git en 2005 para gestionar el código fuente del kernel de Linux y para abordar las limitaciones de otros sistemas de control de versiones que estaba utilizando en ese momento. Git se destacó rápidamente como un sistema de control de versiones distribuido eficiente y se ha convertido en una herramienta fundamental en el desarrollo de software, utilizada en proyectos de todos los tamaños en todo el mundo.

6-Cuales son los comandos más esenciales de Git?

Git tiene una amplia variedad de comandos que permiten a los desarrolladores realizar diversas operaciones en sus repositorios. algunos de los comandos más esenciales de Git:

Configuración Inicial:

1. git init`: Inicializa un nuevo repositorio Git.

2. git config`: Configura opciones específicas del usuario, como nombre y correo electrónico.

Trabajo con Repositorios Remotos:

3. `git clone': Clona un repositorio existente en un nuevo directorio.

4. `git remote`: Muestra los repositorios remotos conectados.

5. `git pull`: Obtiene cambios del repositorio remoto y los fusiona en la rama local.

6. `git push`: Envía cambios locales al repositorio remoto.

Realización de Cambios:

7. `git add`: Agrega cambios al área de preparación (staging).

8. `git commit`: Realiza una confirmación con los cambios agregados en el área de preparación.

9. `git status`: Muestra el estado actual del repositorio.

10. `git diff`: Muestra las diferencias entre los cambios en el directorio de trabajo y el área de preparación.

Ramas:

11. `git branch`:Lista, crea o elimina ramas.

12. `git checkout`: Cambia entre ramas o puntos específicos del historial.

13.`git merge`: Fusiona los cambios de una rama en otra.

Historial:

14. `git log`: Muestra el historial de confirmaciones.

15. `git show`:Muestra información sobre un commit específico.

Deshacer Cambios:

16. `git revert`: Crea una nueva confirmación que deshace los cambios de una confirmación anterior.

17. `git reset`: Deshace cambios locales y puede cambiar la posición de la cabeza.

Etiquetas (Tags):

18. `git tag`: Lista, crea o elimina etiquetas para marcar puntos específicos en la historia.

Trabajo con Submódulos:

19. `git submodule`: Agrega, actualiza o clona submódulos en el repositorio.

Estos son solo algunos de los comandos más esenciales de Git. La combinación de estos comandos proporciona una base sólida para gestionar eficazmente un repositorio Git y participar en flujos de trabajo colaborativos.

7-Que es git Flow?

Git Flow es un conjunto de reglas y convenciones para el uso de Git que establece un flujo de trabajo específico para el desarrollo de software. Fue propuesto por Vincent Driessen y se ha convertido en un modelo popular para organizar el desarrollo de proyectos en equipos colaborativos.

El flujo de trabajo de Git Flow incluye varias ramas y define cómo se deben gestionar y fusionar estas ramas. Las ramas principales en Git Flow son:

`master` (o `main`)

develop

Git Flow proporciona un conjunto de comandos y una estructura clara para la gestión de estas ramas, lo que facilita la colaboración en equipos y la administración del ciclo de vida del software.

Es importante mencionar que aunque Git Flow ha sido muy popular, no es una característica integrada en Git por defecto. Los desarrolladores necesitan instalar y seguir las convenciones de Git Flow para adoptar este modelo de desarrollo. Además, en la actualidad, hay otras metodologías y flujos de trabajo, como GitHub Flow y GitLab Flow, que también ofrecen enfoques eficientes para la gestión de ramas en proyectos Git. La elección entre estas metodologías dependerá de las necesidades y preferencias específicas de cada equipo de desarrollo.

8-Que es trunk based development ?

Trunk Based Development es una metodología de desarrollo de software que se centra en mantener una única rama principal (trunk o rama principal) como el punto central del desarrollo. A diferencia de otros modelos de ramificación más complejos, como Git Flow, Trunk Based Development aboga por una rama principal que sirve como línea de desarrollo principal y que se utiliza para integrar continuamente el trabajo de los desarrolladores.