NOMBRE:

Michael Gabriel Feliciano

MATRICULA:

2022-1325

ASIGNATURA:

Programación III

TEMA:

Tarea 3

DOCENTE:

Kelyn Tejada Belliard

FECHA:

4/3/2025

Contenido

Preguntas

1. ¿Qué es Git?

Git es un proyecto de código abierto maduro y con un mantenimiento activo que desarrolló originalmente Linus Torvalds, el famoso creador del kernel del sistema operativo Linux, en 2005. Un asombroso número de proyectos de software dependen de Git para el control de versiones, incluidos proyectos comerciales y de código abierto. Los desarrolladores que han trabajado con Git cuentan con una buena representación en la base de talentos disponibles para el desarrollo de software, y este sistema funciona a la perfección en una amplia variedad de sistemas operativos e IDE (entornos de desarrollo integrados).

Git, que presenta una arquitectura distribuida, es un ejemplo de DVCS (sistema de control de versiones distribuido, por sus siglas en inglés). En lugar de tener un único espacio para todo el historial de versiones del software, como sucede de manera habitual en los sistemas de control de versiones antaño populares, como CVS o Subversion (también conocido como SVN), en Git, la copia de trabajo del código de cada desarrollador es también un repositorio que puede albergar el historial completo de todos los cambios.

2. ¿Para qué sirve el comando git init?

El comando abre el Inicializar repositorio Git cuadro de diálogo para inicializar un local repositorio Git para el abierto CODESYS proyecto. Durante la inicialización, el proyecto se importa a Git, lo que crea un repositorio local y el índice para el proyecto (almacenamiento de proyectos Git). Como resultado, se crea el directorio de trabajo de Git para el proyecto.

3. ¿Qué es una rama en Git?

En términos técnicos, una rama en Git es un puntero que señala un commit específico dentro del historial. Desde ahí, puedes crear nuevos commits y desarrollar funcionalidades de manera aislada del resto del proyecto.

La verdad es que trabajar con ramas en Git es muy útil, ya que no solo te permite mantener el orden en tu código, sino que también facilita la colaboración entre varios desarrolladores y, por tanto, aumenta la productividad del equipo.

4. ¿Cómo saber en cuál rama estoy trabajando?

Para saber en que rama estas trabajando usamos el siguiente comando: git branch, este nos mostrara una lista de todas las ramas locales y en la rama que estamos trabajando actualmente.

5. ¿Quién creó Git?

Git fue creado originalmente por Linus Torvalds para el control de versiones durante el desarrollo del kernel de Linux. Torvalds quería un sistema distribuido que pudiera usar como BitKeeper, pero ninguno de los sistemas gratuitos disponibles satisfacía sus necesidades. Citó el ejemplo de un sistema de gestión de control de versiones que necesitaba 30 segundos para aplicar un parche y actualizar todos los metadatos asociados, y señaló que esto no se adaptaría a las necesidades del desarrollo del kernel de Linux, donde la sincronización con otros mantenedores podría requerir 250 acciones de este tipo a la vez.

6. ¿Cuáles son los comandos esenciales de Git?

Algunos de los comandos mas esenciales de Git:

-git add
Mueve los cambios del directorio de trabajo al área del entorno de ensayo. Así puedes preparar una instantánea antes de confirmar en el historial oficial.

-git checkout
Además de extraer las confirmaciones y las revisiones de archivos antiguas, git checkout también sirve para navegar por las ramas existentes. Combinado con los comandos básicos de Git, es una forma de trabajar en una línea de desarrollo concreta.

-git clean
Elimina los archivos sin seguimiento de tu directorio de trabajo. Es la contraparte lógica de git reset, que normalmente solo funciona en archivos con seguimiento.

-git clone
Crea una copia de un repositorio de Git existente. La clonación es la forma más habitual de que los desarrolladores obtengan una copia de trabajo de un repositorio central.

-git commit
Confirma la instantánea preparada en el historial del proyecto. En combinación con git add, define el flujo de trabajo básico de todos los usuarios de Git.

-git commit --amend
Pasar la marca --amend a git commit permite modificar la confirmación más reciente. Es muy práctico si olvidas preparar un archivo u omites información importante en el mensaje de confirmación.

-git config
Este comando va bien para establecer las opciones de configuración para instalar Git. Normalmente, solo es necesario usarlo inmediatamente después de instalar Git en un nuevo equipo de desarrollo.

-git fetch
Con este comando, se descarga una rama de otro repositorio junto con todas sus confirmaciones y archivos asociados. Sin embargo, no intenta integrar nada en el repositorio local. Esto te permite inspeccionar los cambios antes de fusionarlos en tu proyecto.

-git init
Inicializa un nuevo repositorio de Git. Si quieres poner un proyecto bajo un control de revisiones, este es el primer comando que debes aprender.

-git log
Permite explorar las revisiones anteriores de un proyecto. Proporciona varias opciones de formato para mostrar las instantáneas confirmadas.

-git pull
Este comando es la versión automatizada de git fetch. Descarga una rama de un repositorio remoto e inmediatamente la fusiona en la rama actual. Este es el equivalente en Git de svn update.

-git push
Enviar (push) es lo opuesto a recuperar (fetch), con algunas salvedades. Permite mover una o varias ramas a otro repositorio, lo que es una buena forma de publicar contribuciones. Es como svn commit, pero envía una serie de confirmaciones en lugar de un solo conjunto de cambios.

-git rebase
Un cambio de base con git rebase permite mover las ramas, lo que ayuda a evitar confirmaciones de fusión innecesarias. El historial lineal resultante suele ser mucho más fácil de entender y explorar.

7. ¿Qué es Git Flow?

Gitflow es un modelo de utilización de ramas para Git y que fue creado en 2010 por Vincent Driessen.

Características:

- Paralelización: Podemos trabajar en varias características a la vez y que no interfieran entre ellas. Podremos incluso, dejar un trabajo a medias, centrarnos en otro y volver cuando sea necesario.

- Colaboración: De la misma forma, varios desarrolladores van a poder estar trabajando en paralelo en varias características del software final sin interferirse.

- A prueba de fallos: Con Gitflow vamos a poder separar el trabajo que estamos realizando en este mismo momento del trabajo ya finalizado. De esta manera, podemos tener trabajo en curso que no funcione sin afectar al desarrollo de la aplicación.

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

El desarrollo basado en troncos garantiza que los equipos publiquen el código de forma rápida y coherente. A continuación, se presenta una lista de ejercicios y prácticas que te ayudarán a perfeccionar el ritmo de trabajo de tu equipo y a desarrollar un calendario de publicaciones optimizado.

Bibliografía

- https://www.excentia.es/
- https://gfourmis.co/gitflow-sin-morir-en-el-intento/
- https://www.atlassian.com/
- https://content.helpme-codesys.com/
- https://git-scm.co
- https://keepcoding.io/