Estudiante:

Hansell Bonilla Parra

Matrícula:

2023-0220

Profesor:

Kelyn Tejada Belliard

Materia:

Programacion III

Tarea 3:

Investigación individual

y Git Flow

INDICE

Introducción

A continuación, este cuestionario está diseñado para explorar los conceptos fundamentales de Git, una de las herramientas de control de versiones más utilizadas en el desarrollo de software. A través de estas preguntas, aprenderemos qué es Git, cómo funciona y para qué sirve en la gestión de proyectos colaborativos. También se revisarán algunos comandos esenciales, como git init, y el concepto de ramas, que permite trabajar en diferentes versiones de un proyecto sin afectar el desarrollo principal.

Además, se incluyen temas avanzados, como Git Flow y Trunk-Based Development, dos enfoques populares para organizar el flujo de trabajo en equipos de desarrollo. Con esta guía, busco proporcionar una base sólida para entender y manejar Git de manera efectiva, permitiendo a los estudiantes aplicar estos conocimientos en proyectos reales.

Desarrolla el siguiente Cuestionario

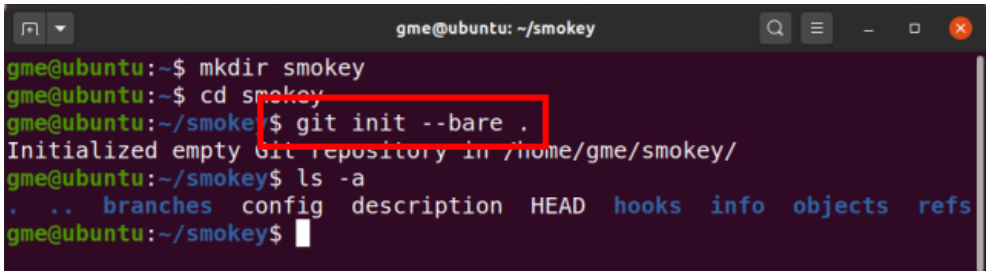
1- ¿Qué es Git?

Git es un sistema de control de versiones distribuido y actualmente es el estándar mundial. Un clon local de tu proyecto es un repositorio de control de versiones completo, estos repositorios locales plenamente funcionales te permiten trabajar sin conexión o de manera remota con muchísima facilidad. Los desarrolladores confirman su trabajo de manera local y luego sincronizan la copia del repositorio con la del servidor.

Cada vez que se guarda el trabajo, Git crea una confirmación, esta es básicamente una instantánea de todos los archivos en un momento dado. Si un archivo no ha cambiado de una confirmación a la siguiente pues Git utiliza el archivo anteriormente almacenado.

Git es sin duda el estándar mundial debido a su eficacia, imaginemos que un equipo de desarrollo de software que trabaja en un proyecto grande, estos pueden usar Git para dividir el proyecto en varias partes, como el front-end y el back-end. Cada desarrollador puede trabajar en su propia copia del repositorio y luego combinar sus cambios en la rama principal teniendo un trabajo compacto y con buenos tiempos de entrega.

2- ¿Para que funciona el comando Git init?



El comando Git init sirve para inicializar (como dice su nombre) un repositorio de Git en un directorio o carpeta vacia, o uno que no este siendo registrado por Git.

Este comando lo utilizamos solamente al configurar un repositorio nuevo por primera vez. Cuando lo ejecutamos, se crea un subdirectorio .git en el directorio que seleccionamos y una nueva rama principal.

Ejemplo: Si tienes una carpeta llamada proyecto-nuevo y deseas comenzar a versionarla. Abres una terminal, navegas a proyecto-nuevo y ejecutas:

git init

así creando el subdirectorio .git del que hablamos donde se almacenarán todas las configuraciones y datos de seguimiento del repositorio.

3- ¿Qué es una rama?

Una rama en Git es una línea independiente de desarrollo dentro del repositorio, es una versión del código donde puedes realizar cambios sin afectar la rama principal o main.

De hecho, la rama por defecto de Git es la rama master. Con la primera confirmación de cambios que realicemos, se creará esta rama principal master apuntando a dicha confirmación.

Un ejemplo perfecto es, si estás trabajando en una nueva función de "Registro de Usuarios", puedes crear una rama llamada feature/registro-usuarios. Esto te permite trabajar en esta característica sin interferir con el resto del código en la rama main.

4- ¿Cómo saber en cual rama estoy?

```
PS C:\Users\Hans\Desktop\tarea3\tarea3_gitflow> git branch
  feature/efectos_visuales
* main
  qa
```

Para saber en cual rama estas actualmente,se necesita abrir una terminal como lo es Git bash o CMD y ejecutamos el comando de “git branch”, esto te mostrará la rama en la que estas con un asterisco junto a ella.

También se puede usar el comando ‘‘git status’’ para saber en qué rama estás.

Si tomamos el ejemplo de la pregunta anterior sobre la nueva función de "Registro de Usuarios", pues al usar git branch o status la salida dirá:

*** feature/registro-usuarios**
main

5- ¿Quién creo git?

Git fue creado en el 2005 por Linus Torvalds, el creador del sistema operativo Linux y surgió por la necesidad de encontrar una solución para el gestionar el código fuente del proyecto del Kernel de Linux. En ese momento el proyecto de Linux necesitaba un sistema de control de versiones que fuera rápido, distribuido y que permitiera la colaboración entre múltiples desarrolladores de manera eficiente.

6- ¿Cuáles son los comandos más esenciales de Git?

Algunos de los comandos más esenciales de Git son:

git init: Crea un nuevo repositorio de Git o reinicia uno existente.

git clone: Crea una copia de un repositorio de Git existente.

git add: Agrega archivos al área de preparación para que Git pueda realizar un

seguimiento de los cambios.

git commit: Guarda los cambios en local y crea un punto de control.

git status: Muestra el estado de la rama actual, incluyendo los cambios que se deben confirmar, enviar o recibir.

git push: Envía los commits locales al repositorio remoto.

git pull: Descarga y fusiona los cambios remotos en el repositorio local.

git branch: Muestra una lista de las ramas del repositorio y resalta la rama actual.

git clean: Elimina los archivos sin seguimiento de tu directorio de trabajo.

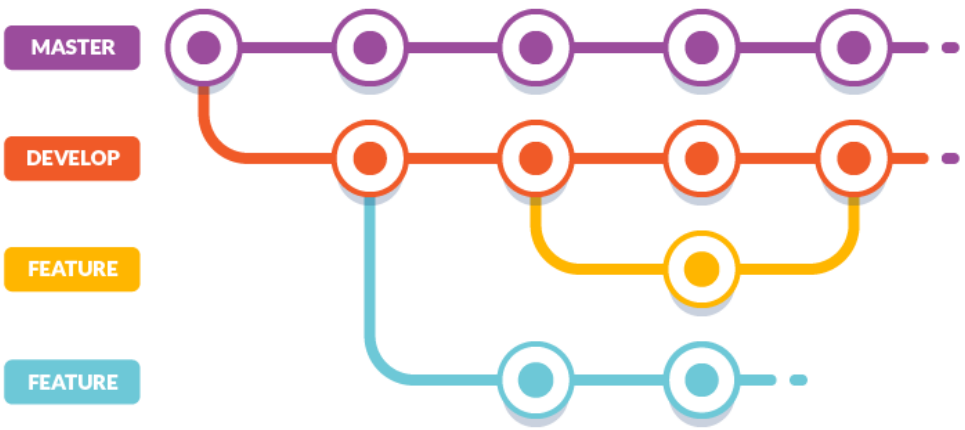
git checkout: Cambia a la rama especificada.

git log: Muestra el historial de commits del proyecto.

git merge: Es una forma eficaz de integrar los cambios de ramas divergentes. Después de bifurcar el historial del proyecto con git branch, git merge permite unirlo de nuevo.

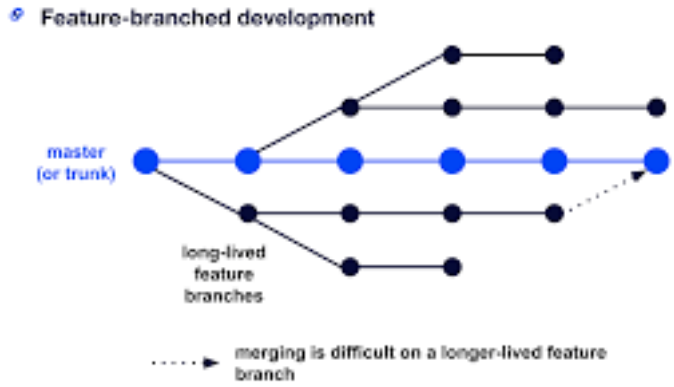
git remote: Es un comando útil para administrar conexiones remotas.

7- ¿Que es git Flow?



Es un flujo de trabajo basado en Git que brinda un mayor control y organización en el proceso de integración continua. Básicamente es una metodología que define una estructura clara de ramas para proyectos, ayudando a los equipos a organizar el desarrollo y las versiones. Incluye ramas como **main** para producción, **develop** para desarrollo, **feature** para nuevas funcionalidades, **release** para preparaciones de lanzamientos, y **hotfix** para correcciones urgentes.
Cuando se quiere desarrollar una nueva función, se crea una rama **feature**, se realizan los cambios y luego se fusiona de vuelta en **develop**. Al estar lista la versión, se pasa a **release**, y luego de ser probada, se fusiona en **main**.

8- ¿Que es trunk based development?



Trunk-based development es un enfoque en el cual todos los desarrolladores trabajan y realizan commits en una sola rama principal (llamada "trunk" o "main"). En lugar de crear múltiples ramas de desarrollo, los desarrolladores integran sus cambios directamente en main, lo que facilita la integración continua y reduce el riesgo de conflictos al final.

Por ejemplo, un equipo que usa trunk-based development, todos los desarrolladores colaboran en la misma rama main y realizan pequeños commits frecuentemente. Este enfoque es útil en proyectos donde se requiere una alta frecuencia de integración y lanzamientos rápidos.

El TBD es una práctica que se está volviendo crucial para las empresas en el mundo digital actual. Algunas de sus ventajas son: Mejora la calidad del software, Agiliza la colaboración, Permite una integración continua, Reduce la frecuencia de los conflictos de fusión, Fomenta la propiedad colectiva del código.

Para que el TBD funcione, se deben seguir algunas prácticas:

- Agregar pruebas automatizadas y monitoreo de la cobertura de código
- Ejecutar pruebas automatizadas antes y después de cada cambio en el tronco
- Fijar inmediatamente cualquier cambio que rompa el tronco
- Utilizar feature flags para encapsular cambios que aún no están terminados

Bibliografía

Carolina, C (2023, agosto, 29). *Descubriendo Git: Características y Ventajas. Blog de Kranio*.

- *https://www.kranio.io/blog/descubriendo-git-caracteristicas-y-ventajas*

Oleh, Subotin (2024, Jul). *Essential Git Commands*. Codefinity Blog.

- https://codefinity.com/blog/Essential-Git-Commands

Leninmhs. (2023, nov, 12) *La historia de Git: el sistema de control de versiones que cambió el mundo de la programación*.

- *https://leninmhs.com/historia-de-git/*

Plattinux (2016, jul, 31*) Linus Torvalds sobre GIT*.

- *https://www.youtube.com/watch?v=iNFtX2ctExM*