Carrera:
Tecnología en Desarrollo de Software

Asignatura:
Programación lll

Docente:
Kelyn Tejada Belliard

Asunto:

Tarea 3

Estudiante:
Robert Alexander Machuca Ramírez (2022-0396)

Periodo Académico:
2023-C2

Fecha de Entrega:
29/07/2023

Desarrolla el siguiente Cuestionario

1- ¿Qué es Git?

Git es un sistema de control de versiones distribuido, diseñado para rastrear cambios en archivos y coordinar el trabajo entre múltiples personas en un proyecto de desarrollo de software. Permite a los desarrolladores mantener un historial completo de cambios realizados en el código fuente, lo que facilita la colaboración, la detección y resolución de conflictos, y la administración de diferentes versiones del proyecto.

2- ¿Para que funciona el comando Git init?

El comando git init se utiliza para inicializar un nuevo repositorio Git en un directorio vacío o en un proyecto existente. Al ejecutar este comando, Git crea una estructura de directorios y archivos ocultos en el directorio del proyecto para comenzar a rastrear los cambios. Después de ejecutar git init, el directorio se convierte en un repositorio Git local donde se puede realizar el seguimiento de versiones y realizar operaciones de control de versiones.

3- ¿Que es una rama?

En Git, una rama (branch en inglés) es una línea de desarrollo independiente que permite a los desarrolladores trabajar en características, correcciones de errores o experimentos sin afectar directamente la rama principal del proyecto, llamada comúnmente "master" o "main". Cada rama tiene su historial de cambios y puede fusionarse con otras ramas para combinar el trabajo realizado en ellas.

4- ¿Como saber en que rama estoy?

Para saber en qué rama estamos ubicados actualmente en Git, podemos usar el comando git branch. Al ejecutar este comando, se mostrará una lista de todas las

ramas presentes en el repositorio, y el nombre de la rama actual estará resaltado con un asterisco (*).

5- ¿Quién creo git?

Git fue creado por Linus Torvalds, el mismo desarrollador que inició el proyecto del kernel de Linux. Inicialmente, lo creó para gestionar el desarrollo del kernel de Linux, pero posteriormente se convirtió en una herramienta ampliamente utilizada en el mundo del desarrollo de software debido a su eficiencia y capacidad para manejar proyectos de cualquier tamaño.

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

- **git clone:** Clona un repositorio existente en un nuevo directorio.
- **git add:** Agrega cambios en archivos al área de preparación (staging area).
- **git commit:** Crea un nuevo commit con los cambios preparados en el área de preparación.
- **git push:** Envía los cambios locales al repositorio remoto.
- **git pull:** Obtiene los cambios del repositorio remoto y los fusiona con la rama actual.
- **git branch:** Muestra la lista de ramas y resalta la rama actual.
- **git merge:** Fusiona una rama específica con la rama actual.
- **git status:** Muestra el estado actual del repositorio.
- **git log:** Muestra el historial de commits.
- **git checkout:** Permite cambiar de rama o restaurar archivos a una versión específica.

7- ¿Qué es git Flow?

Git Flow es una metodología de desarrollo de software que define una serie de reglas y prácticas para trabajar con Git y ramas. Fue popularizada por Vincent Driessen y

proporciona un modelo sólido para proyectos de desarrollo de software que requieren un flujo de trabajo estructurado. La metodología Git Flow propone el uso de diferentes ramas para diferentes propósitos, como "develop" para el desarrollo en curso, "feature" para nuevas características, "release" para preparar versiones, y "hotfix" para corregir errores en producción.

8- ¿Que es trunk based development?

Trunk Based Development (TBD) es una metodología de desarrollo de software en la que los desarrolladores envían cambios al repositorio compartido (el tronco o "trunk" en inglés) con frecuencia y en pequeñas iteraciones. En lugar de mantener largas ramas de desarrollo, los cambios se integran rápidamente en la rama principal, lo que permite detectar problemas y conflictos temprano en el proceso de desarrollo. TBD fomenta una entrega continua y reduce la complejidad asociada con la gestión de múltiples ramas. Es una estrategia que busca mantener el código base en un estado siempre funcional y desplegable, evitando las complicaciones derivadas de fusionar grandes y prolongadas ramas de desarrollo.

2-Desarrolle un ejercicio práctico en Azure DevOps o GitHub con las siguientes características

https://github.com/RobertMachuca12/Prog3Proof