Nombre: Jenssey Morillo

Matricula: 2021-0044

Materia: Programación III

Profesor: Kelyn Tejada Belliard

Asignacion: Tarea 3

1- Desarrolla el siguiente Cuestionario

1- Que es Git?

Git es un sistema de control de versiones distribuido (DVCS, por sus siglas en inglés) que se utiliza 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 de software colaborativo.

Git facilita la colaboración entre desarrolladores al proporcionar un historial detallado de cambios, permitir la gestión eficiente de ramas para trabajar en paralelo y ofrecer la capacidad de deshacer cambios si es necesario. Además, al ser distribuido, cada desarrollador tiene una copia completa del historial del repositorio en su máquina, lo que facilita el trabajo fuera de línea y la colaboración en equipos distribuidos.

2- Para que funciona el comando Git init?

El comando git init crea un nuevo repositorio de Git. Puede utilizarse para convertir un proyecto existente y sin versión en un repositorio de Git, o para inicializar un nuevo repositorio vacío.

3- Que es una rama?

Una rama es una línea independiente de desarrollo dentro del repositorio. Permite trabajar en nuevas características o correcciones sin afectar la rama principal.

4- Como saber es que rama estoy?

Para saber en qué rama estás actualmente en Git, podemos utilizar el comando git branch. Este comando mostrará todas las ramas presentes en nuestro repositorio y resaltará con un asterisco (*) la rama en la que nos encontramos.

5- Quien creo git?

Git fue creado por Linus Torvalds en 2005. Linus, conocido principalmente por ser el creador del kernel de Linux, desarrolló Git para satisfacer las necesidades de colaboración y control de versiones en el desarrollo del kernel de Linux. Este sistema de control de versiones se ha convertido desde entonces en una herramienta ampliamente utilizada en el mundo del desarrollo de software, permitiendo a los programadores llevar un registro de los cambios en archivos de computadora y coordinar el trabajo en repositorios de código compartidos.

6- Cuales son los comandos más esenciales de Git?

Existen varios comandos esenciales en Git que son fundamentales para la mayoría de las operaciones diarias en el control de versiones. Aquí algunos de los comandos más esenciales de Git:

- **git init:** Inicia un nuevo repositorio Git en el directorio actual.
- **git clone [URL]:** Clona un repositorio remoto en tu máquina local.
- **git add [archivo(s)]:** Añade cambios en archivos al área de preparación para el próximo commit.
- **git commit -m "Mensaje del commit":** Guarda los cambios en el repositorio con un mensaje descriptivo.
- **git status:** Muestra el estado actual del directorio de trabajo y del área de preparación.
- **git log:** Muestra el historial de commits.
- **git branch:** Lista las ramas presentes en el repositorio, resaltando la rama actual.
- **git checkout:** Cambia a una rama específica.
- **git merge:** Fusiona una rama con la rama actual.
- **git pull:** Obtiene los cambios más recientes del repositorio remoto y los fusiona con la rama actual.
- **git push:** Sube los cambios locales al repositorio remoto.
- **git remote -v:** Muestra las URL de los repositorios remotos vinculados.
- **git diff:** Muestra las diferencias entre los cambios sin confirmar y el último commit.
- **git tag**: Etiqueta un commit específico para marcar versiones o puntos importantes en el historial.

7- Que es git Flow?

Es un modelo alternativo de creación de ramas en el que se utilizan ramas de función y varias ramas principales. Las ramas de función son ramas dedicadas a desarrollar una serie de funcionalidades, de manera que se pueda agregar funcionalidades nuevas al código sin interferir en las ramas dedicadas a producción.

8- Que es trunk based development?

El Trunk Based Development es una metodología de desarrollo de software que se centra en trabajar principalmente en una única rama principal del repositorio de código. En este enfoque, los desarrolladores realizan cambios directamente en la rama principal y se favorece la integración continua de pequeñas actualizaciones. La idea central es minimizar

la existencia de ramas a largo plazo y promover la entrega frecuente y consistente de nuevas características, mejoras y correcciones.