Asignatura
Programación 3

Profesor
Kelyn Tejada

Asunto
Asignación 3

Nombres
Neftalí Liriano

Matrícula
2022-0437

Fecha
29-07-2023

Desarrolla el siguiente Cuestionario

1-¿Que es Git?

Git es un sistema de control de versiones distribuido que te permite registrar los cambios que haces en tus archivos y volver a versiones anteriores si algo sale mal. Fue diseñado por Linus Torvalds para garantizar la eficiencia y confiabilidad del mantenimiento de versiones de aplicaciones que tienen un gran número de archivos de código fuente.

2-¿Para que funciona el comando Git init?

El comando **git init** se utiliza para inicializar un nuevo repositorio Git en un directorio existente. Cuando ejecutas **git init**, se crea un subdirectorio oculto llamado ".git" que contiene todos los archivos necesarios para que Git funcione. Este directorio ".git" es donde se almacenan todos los metadatos y objetos del repositorio, como el historial de cambios, las referencias a las ramas y las configuraciones.

3-¿Que es una rama?

Una rama en Git es una línea de desarrollo independiente que permite a los desarrolladores trabajar en características, correcciones de errores o mejoras sin afectar directamente el código en la rama principal (generalmente llamada "master" o "main"). Las ramas son útiles para organizar el desarrollo, ya que permiten que diferentes personas trabajen en diferentes aspectos del proyecto al mismo tiempo. Una vez que una rama ha sido probada y se considera estable, puede ser fusionada con otra rama, generalmente con la rama principal.

4-¿Como saber es que rama estoy?

Para saber en qué rama estás actualmente en Git, puedes usar el comando **git branch**. Este comando te mostrará todas las ramas disponibles en el repositorio y marcará con un asterisco (*) la rama en la que te encuentras actualmente.

5-¿Quién creo git?

Git fue creado por Linus Torvalds en 2005. Linus es ampliamente conocido por ser el creador del kernel de Linux, y Git fue desarrollado originalmente para facilitar la colaboración en el desarrollo del kernel de Linux. Desde entonces, Git se ha convertido en uno de los sistemas de control de versiones más populares y ampliamente utilizados en el mundo del desarrollo de software.

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

- **git clone**: Clona (descarga) un repositorio existente en un nuevo directorio.
- **git add**: Agrega cambios al área de preparación (staging area) para ser incluidos en el próximo commit.
- **git commit**: Crea un nuevo commit con los cambios agregados en el área de preparación.
- **git status**: Muestra el estado actual del repositorio, incluyendo archivos modificados, añadidos o eliminados.
- **git log**: Muestra el historial de commits realizados en el repositorio.
- **git push**: Sube los cambios locales a un repositorio remoto.
- **git pull**: Descarga los cambios desde un repositorio remoto y los fusiona con la rama actual.
- **git branch**: Muestra las ramas existentes y crea nuevas ramas.
- **git merge**: Fusiona una rama con otra rama actual.
- **git checkout**: Cambia entre ramas o restaura archivos desde commits anteriores.

7- ¿Que es git Flow?

Git Flow es un conjunto de prácticas y convenciones para utilizar Git de una manera estructurada y organizada. Fue propuesto por Vincent Driessen en 2010 y se ha convertido en una metodología popular para el desarrollo de software que utiliza Git como sistema de control de versiones. Git Flow define un modelo de ramificación concreto, donde diferentes tipos de ramas tienen propósitos específicos, como ramas para nuevas características, ramas para versiones de producción, etc. Esto ayuda a mantener un flujo de trabajo ordenado en proyectos de desarrollo colaborativo.

8- ¿Que es trunk based development ?

Trunk Based Development (TBD) es una metodología de desarrollo de software que se basa en mantener una única rama principal (trunk) como línea de desarrollo principal del proyecto. En lugar de utilizar muchas ramas durante el desarrollo, TBD enfatiza en mantener la rama principal siempre en un estado estable, de forma que todos los cambios y características se integran directamente en esa rama principal.

El objetivo de Trunk Based Development es fomentar la integración continua, acelerar la entrega de cambios y reducir la complejidad que puede surgir al trabajar con múltiples ramas.