NOMBRE Y MATRÍCULA:

Héctor Gabriel Medina García 2022-2045

MATERIA:

Programación 3

FECHA DE ENTREGA:

29/07/2024

TAREA #3 (CUESTIONARIO)

1. ¿Qué es Git?

Git es un sistema de control de versiones distribuido, creado para manejar cualquier cosa, desde pequeños a muy grandes proyectos, con velocidad y eficiencia. A diferencia de los sistemas de control de versiones centralizados (como SVN y CVS), Git utiliza un modelo distribuido. Esto significa que cada desarrollador tiene una copia completa del historial del proyecto, lo que permite trabajar en paralelo sin necesidad de una conexión constante con un servidor central.

Git maneja datos de una manera única, guardando el estado del proyecto como un conjunto de instantáneas (snapshots) en lugar de un cambio lineal de archivos. Esta estructura permite operaciones muy rápidas y eficientes, incluso en grandes proyectos con muchos colaboradores. Además, Git facilita la colaboración a través de servicios como GitHub y GitLab, donde los desarrolladores pueden compartir sus repositorios, contribuir a proyectos de código abierto y realizar revisiones de código.

2. ¿Cuál es el propósito del comando *git init* en Git?

El comando *git init* se utiliza para crear un nuevo repositorio Git. Este comando se ejecuta en el directorio de un proyecto y crea un subdirectorio *.git* que contiene todos los archivos necesarios para el control de versiones. Esto incluye objetos, referencias y configuraciones. *git init* es el primer paso para empezar a usar Git en un proyecto nuevo o existente, y esencialmente convierte un directorio ordinario en un repositorio Git, permitiendo así el seguimiento de los cambios.

Al crear un repositorio, puedes empezar a utilizar otros comandos de Git para añadir archivos (*git add*), realizar commits (*git commit*), crear ramas (*git branch*), entre otros.

3. ¿Qué representa una rama en Git y cómo se utiliza?

Una rama en Git es una línea de desarrollo independiente que permite a los desarrolladores trabajar en nuevas funcionalidades, correcciones de errores o cualquier otro tipo de cambio de manera aislada del resto del proyecto. Cada rama en Git es un puntero a un commit específico. La rama principal, o main (anteriormente llamada master), es generalmente la rama donde se encuentra el código de producción estable.

Los desarrolladores pueden crear nuevas ramas para trabajar en características específicas sin afectar el código estable en main. Una vez que una característica está completa y ha sido probada, se puede fusionar de vuelta a la rama principal usando *git merge*. Las ramas son una herramienta clave para la gestión de proyectos y permiten un flujo de trabajo ordenado y seguro, ya que los cambios se pueden desarrollar y revisar de manera aislada.

4. ¿Cómo puedo determinar en qué rama estoy actualmente en Git?

Para saber en qué rama estamos trabajando, podemos usar el comando *git branch*, que mostrará una lista de todas las ramas locales y destacará la rama actual con un asterisco (*). Otro comando útil es *git status*, que también muestra la rama actual junto con otros detalles como los archivos modificados y el estado del área de

preparación. Estos comandos son esenciales para entender el contexto del trabajo y evitar confusiones al desarrollar múltiples características o corregir errores en paralelo.

5. ¿Quién es la persona responsable de la creación de Git y cuándo fue desarrollado?

Git fue creado por Linus Torvalds, el creador del kernel de Linux, en 2005. Torvalds desarrolló Git después de que la comunidad de desarrollo del kernel de Linux necesitara una nueva herramienta de control de versiones, ya que el sistema que utilizaban en ese momento (BitKeeper) dejó de estar disponible de manera gratuita. Git fue diseñado con tres objetivos principales en mente: velocidad, simplicidad en el diseño y fuerte soporte para ramas (branching). Desde su creación, Git ha sido adoptado ampliamente en la industria del software y se ha convertido en el estándar de facto para el control de versiones.

6. ¿Cuáles son algunos de los comandos esenciales de Git y para qué se utilizan?

git init: Inicializa un nuevo repositorio Git.

git clone: Clona un repositorio existente en una nueva carpeta, descargando todo su historial.

git add: Añade cambios en el directorio de trabajo al área de preparación (staging area), preparándolos para el próximo commit.

git commit: Guarda los cambios preparados en el historial del proyecto. Cada commit tiene un mensaje que describe el cambio.

git status: Muestra el estado actual del directorio de trabajo y el área de preparación.

git push: Envía los commits locales a un repositorio remoto, actualizando la rama correspondiente.

git pull: Descarga cambios desde un repositorio remoto y los fusiona en la rama actual.

git branch: Muestra una lista de ramas locales o crea una nueva rama.

git merge: Fusiona una rama con otra, incorporando los cambios de la rama fuente en la rama destino.

git log: Muestra el historial de commits del repositorio.

7. ¿Puedes mencionar algunos de los repositorios de Git más reconocidos y utilizados en la actualidad?

Linux Kernel: El proyecto más conocido que utiliza Git, donde Linus Torvalds trabaja en el kernel de Linux.

React: Una popular biblioteca de JavaScript para construir interfaces de usuario, mantenida por Facebook.

TensorFlow: Un marco de aprendizaje automático de código abierto desarrollado por Google.

Bootstrap: Un framework front-end para desarrollar interfaces de usuario responsive, desarrollado originalmente por Twitter.

Node.js: Un entorno de ejecución de JavaScript en el lado del servidor, basado en el motor V8 de Chrome.

Django: Un popular framework de desarrollo web para Python.

Docker: Una plataforma para desarrollar, enviar y ejecutar aplicaciones en contenedores.

Kubernetes: Un sistema de orquestación de contenedores de código abierto, mantenido por Google.