NOMBRE: Enjher Javier Agüero Ovalles

MATRÍCULA: 2023-1113

MATERIA: Programación III

TEMA: Módulo 5 – Tarea 3

DOCENTE: Kelyn Tejada Belliard

ÍNDICE

INTRODUCCIÓN

Git es uno de los sistemas de control de versiones más populares y utilizados, ofreciendo una variedad de características, funciones y comandos que permiten a los desarrolladores gestionar sus proyectos de manera organizada,

En este trabajo se presente un cuestionario que aborda diferentes tipos de conceptos relacionados con Git, incluyendo su definición, comandos básicos y/o esenciales, ramificación y metodologías como Git Flow y Trunk-Based Development. Cada respuesta se fundamenta en información obtenida de fuentes confiables, especificadas en la bibliografía al final del documento. Además, se incluye un ejercicio práctico donde se pone en práctica la estructura indicada en la clase.

- **Cuestionario sobre Git**

1. ¿Qué es Git?

Git es un sistema de control de versiones distribuido, lo que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan la copia del repositorio con la del servidor. Este paradigma es distinto del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones.

2. ¿Para qué funciona el comando “Git init”?

Este comando crea un repositorio Git vacío, básicamente un directorio .git con subdirectorios para objects refs/heads, refs/tags y archivos de plantilla.

3. ¿Qué es una rama?

Una rama o branch es una versión del código del proyecto sobre el que estás trabajando. Estas ramas ayudan a mantener el orden en el control de versiones y manipular el código de forma segura.

4. ¿Cómo saber en cuál rama estoy?

Puedes usar el comando git branch para ver todas las ramas locales y la que está activa, marcada con un asterisco (*). Alternativamente, el comando git status también muestra la rama actual.

5. ¿Quién creó Git?

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

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

git init: Inicializa un repositorio.

git clone: Realiza una copia idéntica de la última versión de un proyecto en un repositorio y la guarda en tu ordenador.

git branch: Gestiona las ramas.

git checkout: Cambia de una rama a otra, o revisar archivos o commits.

git status: Nos da la información necesaria sobre la rama actual.

git add: Incluye los cambios del o de los archivos en tu siguiente commit.

git commit: Registra los cambios en el repositorio.

git push: Envía tus commits al repositorio remoto.

git pull: Actualiza el repositorio local desde el remoto.

git revert: Crea un nuevo commit que deshace los cambios hechos por un commit anterior específico.

git merge: Fusiona ramas.

7. ¿Qué es Git Flow?

Git Flow es una estrategia popular de ramificación en Git diseñada para simplificar la gestión de lanzamientos, y fue introducida por el desarrollador de software Vincent Driessen en 2010. Fundamentalmente, Git Flow consiste en aislar el trabajo en diferentes tipos de ramas de Git.

8. ¿Qué es Trunk-Based Development?

Trunk-Based Development es una estrategia en Git donde todo el equipo colabora directamente en una rama principal (trunk), como master o main. No existen ramas de larga duración ni se les da mantenimiento; si surge un error en un branch de release, este se corrige en el trunk y se crea un nuevo release en lugar de mantener el branch original. Se espera que los desarrolladores hagan commits al menos una vez al día para fomentar la colaboración y evitar duplicación de esfuerzos, asegurando que cada commit sea funcional, cumpla con la definición de "hecho", tenga las pruebas necesarias y no introduzca errores. Finalmente, el trunk debe estar siempre en un estado óptimo ("verde") y listo para un lanzamiento, lo que requiere un equipo con alta responsabilidad y madurez técnica.

Bibliografía ejercicio teórico

1. Microsoft Learn. (s.f.). *¿Qué es Git?*. Recuperado de https://learn.microsoft.com/es-es/devops/develop/git/what-is-git
2. Git SCM. (s.f.). *git-init Documentation*. Recuperado de https://git-scm.com/docs/git-init
3. Platzi. (s.f.). *¿Qué es un branch (rama) y cómo funciona un merge en Git?*. Recuperado de https://platzi.com/clases/1557-git-github/19947-que-es-un-branch-rama-y-como-funciona-un-merge-en-/
4. Git SCM. (s.f.). *Ramificaciones en Git - Gestión de Ramas*. Recuperado de https://git-scm.com/book/es/v2/Ramificaciones-en-Git-Gesti%C3%B3n-de-Ramas
5. Wikipedia. (s.f.). *Git*. Recuperado de https://es.wikipedia.org/wiki/Git
6. FreeCodeCamp. (2021). *10 comandos de Git que todo desarrollador debería saber*. Recuperado de https://www.freecodecamp.org/espanol/news/10-comandos-de-git-que-todo-desarrollador-deberia-saber/
7. GitKraken. (s.f.). *Git Flow*. Recuperado de https://www.gitkraken.com/learn/git/git-flow
8. Dev.to. (2020). *¿Por qué Trunk-Based Development?*. Recuperado de https://dev.to/marianocodes/por-que-trunk-based-development-i5n