INSTITUTO TECNOLÓGICO DE LAS AMÉRICAS

Tecnología en desarrollo de software

Asignatura

Programación III

Tema

Tarea 3 – Cuestionario

Facilitador/a

Kelyn Tejada Belliard

Nombre

Carlos A. Henríquez C.

Matricula

2020 – 10203

Fecha

26 – 11 – 2022

DESARROLLO

Desarrolla el siguiente cuestionario:

1- **¿Qué es Git?**

Git es un sistema de control de versiones distribuido el cual fue creado tomando en cuenta la confiabilidad y eficiencia de lo que es el mantenimiento de versiones de aplicaciones cuando estas tienen varios archivos de un código fuente. Esto significa que un clon local de los que es el proyecto es un repositorio de control de versiones completo. Además, Git nos brinda las herramientas necesarias para poder desarrollar un trabajo en equipo de manera rápida e inteligente.

2- **¿Para que funciona el comando "git init"?**

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

3- **¿Qué es una rama?**

En Git, una rama es una división del estado del código. Cuando se quiere añadir ya sea una nueva función o la solución de un error, independientemente de su tamaño, se genera lo que es una nueva rama para alojar los cambios realizados, por lo que las ramas ayudan a mantener el orden en el control de versiones y manipular el código de una forma segura.

4- **¿Cómo saber en qué rama estoy?**

Para saber en que rama estoy en Git se debe de ejecutar el comando "git branch" el cual nos mostrara todas las ramas y nos marcara la rama en donde nos encontramos actualmente.

5- **¿Quién creo Git?**

Git fue creado por Linus Torvalds en el 2005.

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

- **git init:** Sirve para la creación de un repositorio.
- **git clone:** Sirve para descargar el código fuente existente desde un repositorio remoto. Básicamente hace una copia idéntica de la última versión de un proyecto en un repositorio y lo guarda en la computadora.
- **git branch:** Muestra todas las ramas y marca la rama en donde nos encontramos actualmente. Además, sirve para crear, enumerar y eliminar ramas.
- **git checkout:** Se utiliza principalmente para cambiar de una rama a otra. También se puede usar para verificar archivos y confirmaciones.
- **git status:** Este comando nos brinda toda la información necesaria sobre la rama actual.
- **git add:** Sirve para incluir los cambios de un archivo o archivos en el próximo commit.
- **git commit:** Sirve para guardar los cambios localmente. Se puede decir que establece un punto de control en el proceso de desarrollo al que se puede volver más tarde si es necesario. Es necesario escribir un mensaje corto con el que se pueda explicar lo que se ha desarrollado o cambiado en el código fuente.
- **git push:** Envía los cambios al servidor remoto. Sube las confirmaciones al repositorio remoto.
- **git pull:** Se utiliza para obtener actualizaciones del repositorio remoto e inmediatamente aplica los últimos cambios en el local.
- **git revert:** Sirve para deshacer los cambios realizados ya sea de forma local o remota.
- **git merge:** Sirve para fusionar una rama con la rama principal.

7- ¿Qué es Git Flow?

Git Flow es un modelo, una estrategia y también un workflow muy utilizado por los equipos de desarrollo y programación el cual fue publicado por primera vez por Vicent Driessen en la página web Nvie donde se popularizo. Este se destaca por ayudar en la organización de las ramas dentro del repositorio y la seguridad de la versión de un código en lo que son los procesos de creación de nuevos features y hotfixes dentro del código.

En este modelo, los desarrolladores crean una rama de función y retrasan su unión con la rama principal del tronco hasta que la función este completa. Estas ramas de función de larga duración requieren más colaboración para la unión y tienen mayor riesgo de desviarse de la rama troncal. También pueden introducir actualizaciones conflictivas.

8- ¿Qué es trunk based development?

El trunk based development o el desarrollo basado en troncos es una practica de control de versiones en la cual los desarrolladores unen pequeñas actualizaciones de forma frecuente en lo que es un tronco o rama principal. Esta se ha convertido en una práctica habitual dentro de lo que son los equipos de DevOps y parte del ciclo de vida de DevOps, ya que simplifica lo que son las fases de unión e integración. Permitiéndoles a los desarrolladores el poder crear ramas de corta duración con pequeñas confirmaciones. A medida que la complejidad del código base y el tamaño del equipo van creciendo, el desarrollo basado en troncos ayuda a mantener el flujo de publicación de la producción.