Programación III

Nombres: Jorge Apolinar

Apellidos: Pérez Pérez

Matricula: 2019-8961

Materia: Programación III

Profesora: Kelyn Tejada Belliard

Institución: Instituto Tecnológico de las Américas (ITLA)

Carrera: Desarrollo de Software

Desarrolla el siguiente cuestionario

1. ¿Qué es 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.

La flexibilidad y popularidad de Git lo convierten en una excelente opción para cualquier equipo. Muchos desarrolladores y graduados universitarios ya saben cómo usar Git. La comunidad de usuarios de Git ha creado recursos para entrenar a los desarrolladores y la popularidad de Git facilita recibir ayuda cuando se necesita.

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

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. La mayoría de los demás comandos de Git no se encuentran disponibles fuera de un repositorio inicializado, por lo que este suele ser el primer comando que se ejecuta en un proyecto nuevo.

Al ejecutar git init, se crea un subdirectorio de .git en el directorio de trabajo actual, que contiene todos los metadatos de Git necesarios para el nuevo repositorio. Estos metadatos incluyen subdirectorios de objetos, referencias y archivos de plantilla.

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

Son nuevos caminos que toma el proyecto. La rama principal se llama máster y es donde está el proyecto que sale a producción. Cada vez que se saca una nueva característica o que se quiere corregir algo se saca una rama, de tal manera que se pueda trabajar en un ambiente aislado. Es una copia exacta del proyecto, pero está separada. De este modo, si algo se rompe en ese proyecto, no comprometes al proyecto original. Si todo va bien, puedes unificar esa rama con el proyecto principal, si va mal, puedes eliminarla sin ningún problema.

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

Para saber qué ramas están disponibles y cuál es el nombre de la rama actual, ejecuta git Branch, git status.

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

Git es un proyecto de código abierto maduro y con un mantenimiento activo que desarrolló originalmente Linus Torvalds, el famoso creador del kernel del sistema operativo Linux, en 2005. Un asombroso número de proyectos de software dependen de Git para el control de versiones, incluidos proyectos comerciales y de código abierto.

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

1. Git clone
Git clone es un comando para descargarte el código fuente existente desde un repositorio remoto (como Github, por ejemplo). En otras palabras, Git clone básicamente realiza una copia idéntica de la última versión de un proyecto en un repositorio y la guarda en tu ordenador.

git clone <https://link-con-nombre-del-repositorio>

2. Git branch
Las ramas (branch) son altamente importantes en el mundo de Git. Usando ramas, varios desarrolladores pueden trabajar en paralelo en el mismo proyecto simultáneamente. Podemos usar el comando git branch para crearlas, listarlas y eliminarlas.

git branch <nombre-de-la-rama>

Visualización de ramas:
git branch
git branch --list

3. Git checkout
Este es también uno de los comandos más utilizados en Git. Para trabajar en una rama, primero tienes que cambiarte a ella. Usaremos **git checkout** principalmente para

cambiarte de una rama a otra. También lo podemos usar para chequear archivos y commits.

git checkout <nombre-de-la-rama>

4. Git status
El comando de git status nos da toda la información necesaria sobre la rama actual.

git status

5. Git add
Cuando creamos, modificamos o eliminamos un archivo, estos cambios suceden en local y no se incluirán en el siguiente commit (a menos que cambiemos la configuración).

git add <archivo>

6. Git commit
Este sea quizás el comando más utilizado de Git. Una vez que se llega a cierto punto en el desarrollo, queremos guardar nuestros cambios (quizás después de una tarea o asunto específico).

git commit -m "mensaje de confirmación"

7. Git push
Después de haber confirmado tus cambios, el siguiente paso que quieres dar es enviar tus cambios al servidor remoto. Git push envía tus commits al repositorio remoto.

git push <nombre-remoto> <nombre-de-tu-rama>

8. Git pull
El comando **git pull** se utiliza para recibir actualizaciones del repositorio remoto. Este comando es una combinación del **git fetch** y del **git merge** lo cual significa que cundo usemos el git pull recogeremos actualizaciones del repositorio remoto (git fetch) e inmediatamente aplicamos estos últimos cambios en local (git merge).

git pull <nombre-remoto>

9. Git revert
A veces, necesitaremos deshacer los cambios que hemos hecho. Hay varias maneras para deshacer nuestros cambios en local y/o en remoto (dependiendo de lo que necesitemos),

pero necesitaremos utilizar cuidadosamente estos comandos para evitar borrados no deseados.

solo necesitamos especificar el código de comprobación que encontrarás junto al commit que queremos deshacer:

git revert 3321844

10. Git merge
Cuando ya hayas completado el desarrollo de tu proyecto en tu rama y todo funcione correctamente, el último paso es fusionar la rama con su rama padre (dev o master). Esto se hace con el comando git merge.

Git merge básicamente integra las características de tu rama con todos los commits realizados a las ramas dev (o master).

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

Algunos de los repositorios de Git más reconocidos y utilizados en la actualidad son:

- GitHub
- GitLab
- Bitbucket