Instituto Tecnológico De Las Américas (ITLA)

Presentación

Nombre:

Anjeisi Ivelisse Acosta Carvajal

Matricula:

2021-1815

Asignatura:

Programación III

Facilitador:

Kelyn Tejada

Tema:

Asignación individual

Fecha:

27-03-2023

1-Desarrolla el siguiente Cuestionario

1-Que es Git?

Git es un sistema de control de versiones de código abierto diseñado para rastrear cambios en archivos y colaborar en proyectos de software de manera eficiente. Fue creado por Linus Torvalds en 2005 y se ha convertido en uno de los sistemas de control de versiones más populares y ampliamente utilizados en la industria del desarrollo de software.

Git permite a los desarrolladores trabajar juntos en un proyecto de manera más eficiente, ya que pueden mantener un registro completo de los cambios que se han realizado en el código y quién los hizo.

Esto facilita la colaboración y la resolución de conflictos en un equipo de desarrollo. Además, Git proporciona herramientas para realizar un seguimiento de diferentes versiones del código y revertir cambios en caso de que algo salga mal.

2-Para que funciona el comando Git init?

El comando "git init" se utiliza para inicializar un nuevo repositorio de Git en un directorio existente. Cuando se ejecuta el comando "git init", Git crea un nuevo subdirectorio oculto llamado ".git" en el directorio actual. Este subdirectorio es donde Git almacena todos los datos y metadatos relacionados con el control de versiones.

Es importante tener en cuenta que la inicialización de un repositorio de Git con "git init" solo se realiza una vez en un directorio. Si ya se ha inicializado un repositorio de Git en un directorio, no es necesario volver a hacerlo a menos que se elimine el subdirectorio ".git" o se quiera comenzar un nuevo repositorio.

4-Que es una rama?

En Git, una rama (branch en inglés) es una línea de desarrollo independiente que se ramifica del tronco principal del repositorio (también conocido como rama principal o "master"). En otras palabras, una rama es una copia virtual de la rama principal del repositorio en la que se pueden hacer cambios sin afectar directamente a la rama principal.

Cada rama de Git tiene su propia historia de confirmaciones (commits), lo que significa que se pueden hacer cambios en una rama sin afectar la rama principal o cualquier otra rama.

Las ramas se utilizan comúnmente en Git para desarrollar nuevas características, corregir errores o experimentar con diferentes enfoques sin interrumpir el trabajo en la rama principal.

Una vez que se completa el trabajo en una rama, se puede combinar (merge) de nuevo en la rama principal para incorporar los cambios. También es posible fusionar varias ramas juntas para consolidar los cambios de varias líneas de desarrollo en una sola rama.

3-Como saber es que rama estoy?

Para saber en qué rama se encuentra actualmente en Git, puede utilizar el comando "git branch" seguido de la opción "-a" (para mostrar todas las ramas) o "-v" (para mostrar información adicional sobre las ramas).

Si solo se desea ver la rama actual resaltada, se puede agregar la opción "--show-current" al comando "git branch", como sigue:

Este comando mostrará el nombre de la rama actual en la que se encuentra actualmente. Además, en la línea de comandos, el nombre de la rama actual a menudo se mostrará como parte del indicador de línea de comandos (por ejemplo, "nombre-de-usuario@nombre-de-equipo nombre-de-rama $").

5-Quien creo git?

Git fue creado por Linus Torvalds, el creador del sistema operativo Linux. Torvalds comenzó a trabajar en Git en 2005 después de que la relación con la comunidad de desarrolladores de Linux se volviera tensa debido a problemas con el sistema de control de versiones anterior que se utilizaba en el proyecto.

Torvalds diseñó Git para ser rápido, escalable y seguro, y lo lanzó como un proyecto de código abierto bajo la Licencia Pública General de GNU (GPL) versión 2.

Desde entonces, Git ha ganado una gran popularidad y se ha convertido en uno de los sistemas de control de versiones más utilizados en la industria del desarrollo de software.

6-Cuales son los comandos más esenciales de Git?

Aquí hay algunos comandos esenciales de Git que son ampliamente utilizados:

git init: inicializa un nuevo repositorio de Git.

git clone: clona un repositorio existente de Git en una nueva ubicación.

git add: agrega archivos al área de preparación (staging) de Git para prepararlos para una confirmación.

git commit: confirma los cambios realizados en el área de preparación y los almacena en el repositorio.

git status: muestra el estado actual de los archivos en el directorio de trabajo y en el área de preparación.

git branch: muestra las ramas existentes en el repositorio y crea nuevas ramas.

git checkout: cambia entre ramas, confirma o deshace confirmaciones y despliega archivos.

git merge: fusiona una rama en otra rama.

git pull: actualiza el repositorio local con los cambios remotos.

git push: envía los cambios locales al repositorio remoto.

7-Que es git Flow?

Git Flow es una metodología de gestión de ramas de Git que se utiliza comúnmente en el desarrollo de software.

Fue creado por Vincent Driessen en 2010 como un modelo de ramificación y flujo de trabajo para proyectos de software más grandes y complejos.

El modelo de Git Flow utiliza dos ramas principales:

Master y Develop.

La rama "master" se utiliza para mantener la versión estable y la rama "develop" se utiliza para integrar el trabajo de las distintas ramas de desarrollo.

Además de estas dos ramas principales, Git Flow utiliza varias ramas adicionales para gestionar las distintas fases del desarrollo.

Estas incluyen:

"**feature**": se utiliza para trabajar en una nueva funcionalidad.

"**release**": se utiliza para preparar una versión para su lanzamiento.

"**hotfix**": se utiliza para corregir problemas críticos en la versión actual en producción.

"**support**": se utiliza para mantener versiones antiguas del software mientras se trabaja en una versión nueva.

8-Que es trunk based development ?

Trunk Based Development (TBD) es una metodología de desarrollo de software que se enfoca en tener una única rama principal (conocida como tronco o "trunk" en inglés) y hacer integraciones frecuentes y pequeñas en ella. En el TBD, se espera que los desarrolladores integren sus cambios en el tronco lo más rápido posible, lo que permite una mayor colaboración y una entrega de código más rápida y frecuente.

A diferencia de otras metodologías de ramificación como Git Flow, que pueden generar ramificaciones complejas y mantenimiento adicional, el TBD se enfoca en mantener una rama principal siempre estable y en condiciones de ser desplegada. Los desarrolladores trabajan en pequeñas tareas y características y las integran directamente en el tronco tan pronto como estén listas, en lugar de trabajar en ramas separadas. Esto ayuda a reducir los conflictos y los problemas de integración que pueden surgir al final de un ciclo de desarrollo más largo.

El TBD se basa en la idea de que los ciclos de retroalimentación rápidos y la entrega continua son esenciales para mantener un ritmo de desarrollo sostenible y de alta calidad. Si bien puede no ser adecuado para todos los proyectos, el TBD se ha vuelto cada vez más popular en entornos de desarrollo ágil y DevOps donde se valora la rapidez, la colaboración y la calidad del código.

2-Desarrolle un ejercicio práctico en Azure Devops o GitHub con las siguientes características.

Entregas:

Link de GitHub donde se desarrollen las siguientes actividades:

- -Crear un proyecto.
- -Utilizar la técnica Git Flow en su proyecto.
- -Proyecto funcional.
- Link del GitHub:
  https://github.com/anjeisi/TiendaAnjeisi