INSTITUTO TECNOLÓGICO DE LAS AMÉRICAS

Docente
Kelyn Tejada

Trabajo
Tarea 3

Asignatura
Programación 3

Grupo
2

Alumno
Carlos Daniel Taveras Liranzo

Matrícula
2021-2021

Fecha de entrega
31-3-2023

Ciclo / Año
Cuatrimestre 1 / 2023

Contenido

Asignación individual

Valor 5 Puntos

Trabajar de forma individual

Desarrolla el siguiente Cuestionario

1-Que 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. (mijacobs, 2023).

“GIT Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.” (de, 2007)

GIT es un sistema que te ayuda a organizar el código de trabajo, el historial y evolución de este. (Pablo, 2019)

2-Para que funciona el comando Git init?

Git init es un comando que se utiliza una sola vez durante la configuración inicial de un repositorio nuevo. Al ejecutar este comando, se creará un nuevo subdirectorio (.git) en tu directorio de trabajo actual. También se creará una nueva rama principal.

Para ejecutar el comando "git init" y crear un nuevo repositorio Git, debes seguir los siguientes pasos:

1. Abre la terminal de tu sistema operativo (puede ser la terminal de Linux, Mac o el símbolo del sistema en Windows).
2. Accede al directorio donde deseas crear el repositorio. Por ejemplo, si deseas crear el repositorio en la carpeta "Documentos", escribe el comando "cd Documentos" en la terminal y presiona Enter.
3. Una vez que te encuentres en el directorio deseado, escribe el comando "git init" y presiona Enter.
4. Git inicializará un nuevo repositorio vacío en el directorio actual.

4-Que 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. (¿Qué Es Branch (Rama) Y Cómo Funciona Un Merge En Git?, 2020).

Una rama Git es simplemente un apuntador móvil apuntando a una de esas confirmaciones. La rama por defecto de Git es la rama master. (de, 2007).

3-Como saber en qué rama estoy?
Para saber en qué rama estás actualmente en Git, puedes usar el siguiente comando en la línea de comandos:

git branch

Este comando te mostrará una lista de todas las ramas locales en tu repositorio de Git, y la rama en la que estás actualmente se destacará con un asterisco (*) al lado de su nombre. Por ejemplo, si estás en la rama "master", verás algo como esto:

* master

development

feature-branch

En este ejemplo, la rama "master" es la rama actualmente activa, ya que tiene un asterisco (*) al lado de su nombre.

5-Quien creo git?
Linus Benedict Torvalds.

6-Cuales son los comandos más esenciales de Git?

Git clone

Git clone es un comando para descargar el código fuente existente desde un repositorio remoto (como GitHub, por ejemplo). En otras palabras, Git clon básicamente hace una copia idéntica de la última versión de un proyecto en un repositorio y la guarda en su computadora.

Hay un par de formas de descargar el código fuente, pero sobre todo prefiero el **clon con https** :

git clone <https://name-of-the-repository-link>

Por ejemplo, si queremos descargar un proyecto de GitHub, todo lo que tenemos que hacer es hacer clic en el botón verde (clonar o descargar), copiar la URL en el cuadro y pegarla después del comando git clone que he mostrado a la derecha. Arriba.

Git branch

Las ramas son muy importantes en el mundo de git. Mediante el uso de ramas, varios desarrolladores pueden trabajar en paralelo en el mismo proyecto simultáneamente. Podemos usar el comando git branch para crear, enumerar y eliminar ramas.

Creando una nueva sucursal: *git branch <branch-name>*

Este comando creará una rama **localmente** . Para insertar la nueva rama en el repositorio remoto, debe usar el siguiente comando: *git push -u <remote> <branch-name>*

Visualización de sucursales: *git branch or git branch --list*

Eliminar una rama: *git branch -d <branch-name>*

Git checkout

Para trabajar en una sucursal, primero debe cambiarse a ella. Usamos **git checkout** principalmente para cambiar de una sucursal a otra. También podemos usarlo para verificar archivos y confirmaciones.

git checkout <name-of-your-branch>

Hay algunos pasos que debe seguir para cambiar con éxito entre sucursales:

- Los cambios en su rama actual deben confirmarse o guardarse antes de cambiar
- La sucursal que desea verificar debe existir en su local

También hay un comando de acceso directo que le permite crear y cambiar a una rama al mismo tiempo:

git checkout -b <name-of-your-branch>

Este comando crea una nueva rama en su local (-b significa rama) y marca la rama como nueva justo después de que se haya creado.

Git status

El comando de estado de Git nos brinda toda la información necesaria sobre la rama actual:

git status

Podemos recopilar información como:

- Si la rama actual está actualizada
- Si hay algo que cometer, empujar o tirar
- Si hay archivos preparados, sin preparar o sin seguimiento
- Si hay archivos creados, modificados o eliminados

Git add

Cuando creamos, modificamos o eliminamos un archivo, estos cambios ocurrirán en nuestro local y no se incluirán en la próxima confirmación (a menos que cambiemos las configuraciones).

Necesitamos usar el comando git add para incluir los cambios de un archivo en nuestra próxima confirmación.

Para agregar un solo archivo: *git add <file>*

Para agregar todo a la vez: *git add -A*

Importante: *el comando git add no cambia el repositorio y los cambios no se guardan hasta que usamos git commit.*

Git commit

Una vez que llegamos a cierto punto en el desarrollo, queremos guardar nuestros cambios (tal vez después de una tarea o problema específico).

Git commit es como establecer un punto de control en el proceso de desarrollo al que puede volver más tarde si es necesario.

También necesitamos escribir un mensaje corto para explicar lo que hemos desarrollado o cambiado en el código fuente.

git commit -m "commit message"

Importante: *Git commit guarda tus cambios solo localmente*

Git push

Después de confirmar sus cambios, lo siguiente que desea hacer es enviar sus cambios al servidor remoto. Git push sube tus confirmaciones al repositorio remoto.

git push <remote> <branch-name>

Sin embargo, si su rama se creó recientemente, también debe cargar la rama con el siguiente comando:

git push --set-upstream <remote> <name-of-your-branch>

o

git push -u origin <branch_name>

Importante*:* Git push solo carga los cambios que están confirmados.

Git pull

El comando **git pull** se usa para obtener actualizaciones del repositorio remoto. Este comando es una combinación de **git fetch** y **git merge** , lo que significa que, cuando usamos git pull, obtiene las actualizaciones del repositorio remoto (git fetch) e inmediatamente aplica los últimos cambios en su local (git merge).

git pull <remote>

Importante**:** Esta operación puede causar conflictos que debe resolver manualmente**.**

Git revert

Una forma más segura de deshacer nuestras confirmaciones es usando git revert . Para ver nuestro historial de confirmaciones, primero debemos usar git log – oneline. Luego, solo necesitamos especificar el código hash junto a nuestro compromiso que nos gustaría deshacer:

git revert 3321844

La ventaja de usar git revert es que no toca el historial de confirmaciones. Esto significa que aún puede ver todas las confirmaciones en su historial, incluso las revertidas.

Git merge

Git merge básicamente integra su rama de características con todas sus confirmaciones en la rama dev (o maestra). Es importante recordar que primero debe estar en la rama específica que desea fusionar con su rama de función.

Por ejemplo, cuando desee fusionar su rama de funciones en la rama de desarrollo:

-**Primero debes cambiar a la rama de desarrollo**: *git checkout dev*

-**Antes de fusionarse, debe actualizar su rama de desarrollo local:** *git fetch*

-**Finalmente, puede fusionar su rama de características en dev:** *git merge <branch-name>*

Sugerencia: asegúrese de que su rama de desarrollo tenga la última versión antes de fusionar sus ramas, de lo contrario, puede enfrentar conflictos u otros problemas no deseados.

7-Que es git Flow?

Git Flow es un modelo de ramificación que se utiliza para organizar el flujo de trabajo en proyectos de software. Este modelo se basa en la idea de tener dos ramas principales: la rama "DEVELOP" y la rama "master". La rama "DEVELOP" es donde se desarrolla el código en su estado actual y la rama "master" es donde se guarda el código que se considera listo para su lanzamiento.

Además de estas dos ramas principales, el modelo de Git Flow también utiliza otras ramas, como "feature branches" y "release branches", para facilitar el desarrollo y la entrega de características específicas y para la preparación de lanzamientos.

- Rama Master: Rama por defecto del repositorio.
- Rama DEVELOP: Rama de desarrollo.
- Rama Features: Nuevas funcionalidades y corregir errores.
- Rama reléase: Último retoque.
- Rama Holfix: Rama para errores en producción.

8-Que es trunk based development ?

Es un enfoque de control de versiones que se centra en trabajar en una sola rama principal, llamada "trunk", en lugar de crear múltiples ramas. En este enfoque, los desarrolladores realizan sus cambios directamente en la rama "trunk" y se utilizan técnicas de integración continua y pruebas automatizadas para garantizar que el código sea estable y esté listo para el lanzamiento en todo momento.

Trunk-Based Development es un enfoque más simplificado en comparación con Git Flow, lo que puede ser beneficioso para proyectos más pequeños o para equipos que prefieren un flujo de trabajo más ágil y rápido.

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.

Enlace al repositorio de GitHub: https://github.com/CarlosDaniel002/Weather-App.git

Webgrafía

de, C. (2007, October 26). software de control de versiones. Wikipedia.org; Wikimedia Foundation, Inc. https://es.wikipedia.org/wiki/Git

mijacobs. (2023, February 9). ¿Qué es Git? - Azure DevOps. Microsoft.com. https://learn.microsoft.com/es-es/devops/develop/git/what-is-git

Atlassian. (2023). Qué es Git | Atlassian Git Tutorial. Atlassian. https://www.atlassian.com/es/git/tutorials/what-is-git

Cem Eygi. (2020, January 19). 10 Git Commands Every Developer Should Know. FreeCodeCamp.org; freeCodeCamp.org. https://www.freecodecamp.org/news/10-important-git-commands-that-every-developer-should-know/