Pasos de la práctica guiada que se realizará en Visual Studio Code:

\*\*Práctica Guiada 4:

Fusión de Ramas y Cambio de Versión en Visual Studio Code\*\*

\*Objetivo:\* Aprender a fusionar una rama en la rama principal utilizando el comando `merge` y luego realizar un cambio de versión hacia atrás utilizando Git en Visual Studio Code.

\*\*Paso 1: Crear una Rama y Realizar Cambios\*\*

1. Abre Visual Studio Code.

2. Abre la carpeta de tu repositorio Git en Visual Studio Code.

3. Abre la terminal integrada en Visual Studio Code.

4. Crea una nueva rama llamada "rama-nueva-funcionalidad" y cámbiate a ella:

```markdown

# Crear una nueva rama.

git branch rama-nueva-funcionalidad

# Cambiar a la nueva rama.

git checkout rama-nueva-funcionalidad

```

5. Realiza cambios en el archivo "notas.txt" (o cualquier archivo de tu elección), agrega contenido y guárdalo.

\*\*Paso 2: Fusionar la Rama en la Rama Principal\*\*

6. Cambia de nuevo a la rama principal (por ejemplo, "main" o "master"):

``` markdown

# Cambiar a la rama principal.

git checkout main

```

7. Fusiona la rama "rama-nueva-funcionalidad" en la rama principal:

``` markdown

# Fusionar la rama "rama-nueva-funcionalidad" en la rama principal.

git merge rama-nueva-funcionalidad

```

8. Observa cómo los cambios de la rama "rama-nueva-funcionalidad" se han incorporado a la rama principal.

\*\*Paso 3: Realizar un Cambio de Versión Atrás\*\*

9. Utiliza el comando `git log` para ver el registro de commits y encontrar el hash (identificador) del commit al que deseas regresar:

``` markdown

# Ver el registro de commits.

git log

```

10. Copia el hash del commit al que deseas regresar.

\*\*Paso 4: Revertir al Commit Anterior\*\*

11. En la terminal de Visual Studio Code, utiliza el comando `git revert` seguido del hash del commit para revertir los cambios y volver a la versión anterior:

``` markdown

# Revertir al commit anterior utilizando su hash.

git revert <hash\_del\_commit>

```

12. Confirma la operación y guarda los cambios.

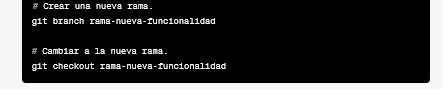
13. Observa cómo los cambios realizados en el commit seleccionado se han revertido y el repositorio ha vuelto a una versión anterior.

Esta práctica guiada se realiza en Visual Studio Code y permite aprender a fusionar ramas con el comando `merge` y a revertir cambios utilizando el comando `revert` en Git. También se practica cómo utilizar el registro de commits (`git log`) para identificar commits específicos a los que desean volver.

**Algoritmo de Pasos para Realizar la Práctica Guiada en Visual Studio Code**

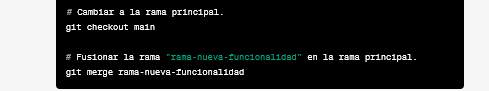
Nota: Asegúrate de que la carpeta de tu repositorio Git esté abierta en Visual Studio Code antes de comenzar.

**Paso 1: Crear una Rama y Realizar Cambios**



Realiza cambios en el archivo "notas.txt" (o cualquier archivo de tu elección), agrega contenido y guárdalo.

**Paso 2: Fusionar la Rama en la Rama Principal**



**Paso 3: Realizar un Cambio de Versión Atrás**



Copia el hash del commit al que deseas regresar.



Confirma la operación y guarda los cambios.

Observa cómo los cambios realizados en el commit seleccionado se han revertido y el repositorio ha vuelto a una versión anterior.

**Fast-Forward (Avance Rápido):**

* **Explicación:** El "fast-forward" es un tipo de fusión que ocurre cuando se fusiona una rama con otra y no hay conflictos entre los cambios. En otras palabras, Git puede avanzar rápidamente desde la punta de la rama actual hasta la punta de la rama que se desea fusionar, ya que no hay ramificaciones ni cambios conflictivos en el camino.
* **Uso:** Se utiliza cuando se desea fusionar una rama en la rama actual y los cambios son lineales y se pueden aplicar sin problemas.

**Fusión Automática:**

* **Explicación:** La fusión automática es un proceso en el que Git intenta fusionar automáticamente dos ramas cuando no hay conflictos entre los cambios. Git detecta las diferencias entre las ramas y aplica los cambios de manera automática, creando un nuevo commit de fusión si es necesario.
* **Uso:** La fusión automática es la forma más común de fusión en Git y se utiliza cuando los cambios en las ramas se pueden fusionar sin conflictos. Se realiza utilizando comandos como git merge y, en muchos casos, Git puede realizarla sin intervención del usuario.

**Fusión Manual:**

* **Explicación:** La fusión manual ocurre cuando Git detecta conflictos entre los cambios en las ramas que se están fusionando. En este caso, Git no puede resolver los conflictos automáticamente y requiere la intervención del usuario para determinar cómo deben resolverse los conflictos.
* **Uso:** La fusión manual se utiliza cuando los cambios en las ramas son conflictivos y deben ser revisados y resueltos por el usuario. Para resolver conflictos, el usuario debe editar manualmente los archivos con conflictos y luego realizar un commit de fusión para finalizar la operación.

En resumen, "fast-forward" es una fusión rápida sin conflictos, la fusión automática ocurre cuando Git puede fusionar automáticamente los cambios y la fusión manual se utiliza cuando se requiere intervención del usuario para resolver conflictos. Estos conceptos son fundamentales para comprender cómo funcionan las operaciones de fusión en Git.

**1. Comando merge:**

* **Uso:** El comando merge se utiliza para fusionar cambios de una rama en otra rama. Es decir, toma los cambios de una rama y los integra en otra, creando un nuevo commit que representa la fusión de ambas ramas.
* **Sintaxis:** El comando básico para realizar una fusión (merge) es:



* Donde <nombre\_de\_la\_rama> es el nombre de la rama que deseas fusionar con la rama actual.
* **Resultado:** Cuando ejecutas un merge, los cambios de la rama especificada se incorporan a la rama actual. Se crea un nuevo commit que combina los cambios de ambas ramas. Si no hay conflictos entre los cambios, la fusión se realiza de manera automática.

**2. Comando revert:**

* **Uso:** El comando revert se utiliza para deshacer un commit específico. En lugar de eliminar un commit, crea un nuevo commit que revierte los cambios realizados en el commit que se desea deshacer.
* **Sintaxis:** El comando básico para realizar una reversión (revert) es:



* git revert <hash\_del\_commit>
* Donde <hash\_del\_commit> es el identificador (hash) del commit que deseas deshacer.
* **Resultado:** Al ejecutar git revert, se crea un nuevo commit que deshace los cambios introducidos por el commit especificado. Esto permite mantener un historial de versiones y es una forma segura de deshacer cambios sin perder el historial.

**3. Comando git log:**

* **Uso:** El comando git log se utiliza para ver el registro de commits en un repositorio Git. Proporciona información detallada sobre cada commit, como el autor, la fecha, el mensaje de commit y el hash (identificador) del commit.
* **Sintaxis:** El comando básico para ver el registro de commits es:



* **Resultado:** Al ejecutar git log, se mostrará una lista de commits en orden cronológico inverso (el commit más reciente primero). Cada entrada en el registro incluirá detalles sobre el commit, lo que te permite rastrear y entender la historia de desarrollo de un proyecto Git.

Espero que estas explicaciones te ayuden a comprender mejor los comandos merge, revert y git log, así como su utilidad en el trabajo con repositorios Git. Si tienes más preguntas o necesitas más detalles, no dudes en preguntar.

Un "hash" en el contexto de la criptografía y la informática se refiere a una función matemática que toma un conjunto de datos (como un archivo, una cadena de texto o cualquier entrada) y produce una cadena de caracteres alfanuméricos de longitud fija que representa de manera única esos datos. Estos valores resultantes se conocen como "hashes".

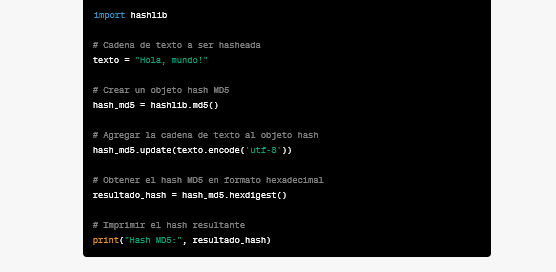
Un ejemplo de hash utilizando Markdown para representar el resultado:

Supongamos que tenemos una cadena de texto "Hola, mundo!" y queremos calcular su hash MD5. El resultado sería el siguiente:



El valor "2ef7bde608ce5404e97d5f042f95f89f1" es el hash MD5 resultante de la cadena de texto "Hola, mundo!" utilizando el algoritmo MD5. Cabe destacar que este es solo un ejemplo y que el hash real de cualquier entrada puede variar.

Otro ejemplo común de función de hash es el algoritmo MD5 (Message Digest Algorithm 5). Aquí hay un ejemplo simple utilizando Python para calcular el hash MD5 de una cadena de texto:



Este código calculará el hash MD5 de la cadena de texto "Hola, mundo!" y mostrará el resultado. Interesante destacar que el hash MD5 produce un valor único para cada entrada, por lo que incluso un pequeño cambio en la entrada generará un hash completamente diferente.

Es importante señalar que, en la actualidad, MD5 se considera débil para aplicaciones de seguridad debido a su susceptibilidad a colisiones y a los ataques de fuerza bruta. En lugar de MD5, se recomienda el uso de algoritmos de hash más seguros como SHA-256 o SHA-3 en aplicaciones de seguridad y criptografía.