**Práctica 04**

**Contenido**

[1](#_heading=h.4d34og8) Creando alias para nuestros comandos 2

[2](#_heading=h.2s8eyo1) Cambios en los archivos 2

[2.1](#_heading=h.17dp8vu) Comparar cambios en archivos con visual studio code 4

[3](#_heading=h.3rdcrjn) Actualizar mensajes de commits y deshacer commits 5

[3.1](#_heading=h.26in1rg) Actualizar el mensaje de un commit 5

[3.2](#_heading=h.lnxbz9) Borrar un commit 5

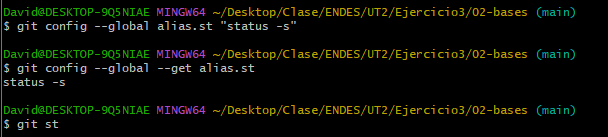
[4](#_heading=h.35nkun2) Preparando un repositorio para viajes en el tiempo 5

1. **Creando alias para nuestros comandos**

Para mostrar el estado de los archivos del repositorio con una descripción corta podemos usar el siguiente comando:

git status -s

Podemos crear un alias para este comando de forma que podamos ejecutarlo escribiendo menos texto:



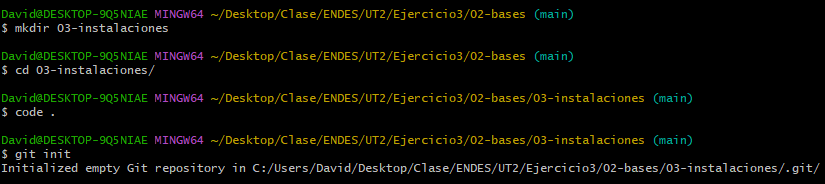
Como vemos para crear un alias usamos la configuración global y ponemos alias. y después del punto ponemos el texto que servirá de alias. A continuación escribimos entre comillas el comando que queremos abreviar sin poner git delante.

1. **Cambios en los archivos**

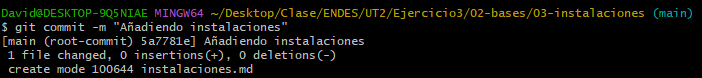
Creamos un nuevo repositorio que se llame 03-instalaciones y lo abrimos en visual studio.

Hacemos el **git init** para inicializar el repositorio y creamos un archivo que se llame instalaciones.md con el siguiente contenido:

Lo agregamos al stage:



Y hacemos commit:



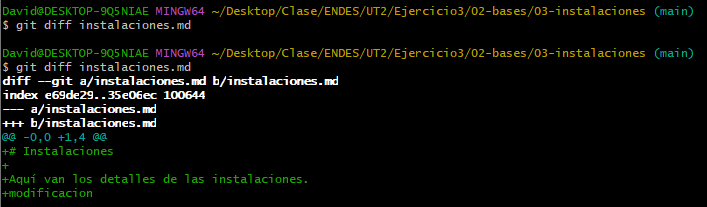
Deberíamos ver algo así en nuestro visual studio code:

Modificamos el archivo instalaciones y lo dejamos así:

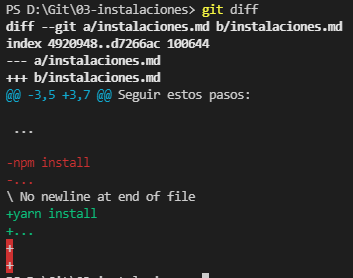
(Hemos modificado una línea y añadido 3 líneas vacías debajo de la segunda línea con …)

Guardamos los cambios, pero no hacemos add ni commit.

Vamos a comparar las modificaciones realizadas en este archivo, para ellos usamos el siguiente comando:

****

La versión a del archivo instalaciones nos dice que tiene menos cosas que la versión b del mismo archivo



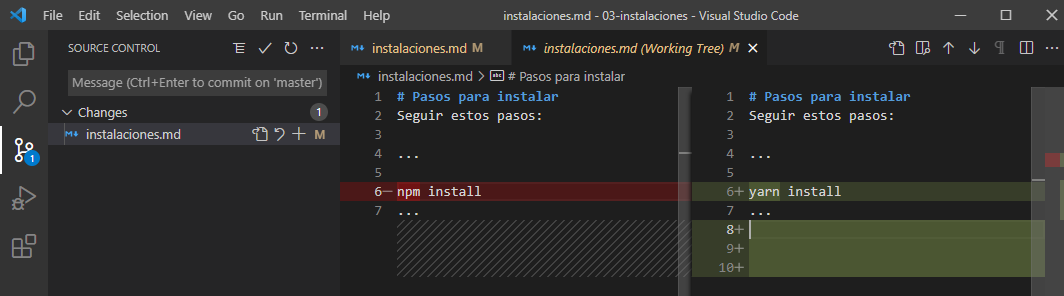
Las líneas en color rojo y precedidas por un - nos indican cosas la primera versión no tiene respecto al segundo y las cosas en verde precedidas por un + nos indican cosas que la segunda versión tiene y la primera no.

El comando git diff nos compara las modificaciones de los archivos que no están en el escenario. Si queremos usar el comando con un archivo que hemos añadido al escenario usaremos lo siguiente:



* 1. **Comparar cambios en archivos con visual studio code**

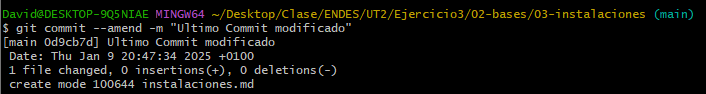
Desde la opción de source control de visual studio podemos comparar las modificaciones realizadas en un archivo de formas más visual:



Hacemos click en el archivo del cual queremos ver las diferencias

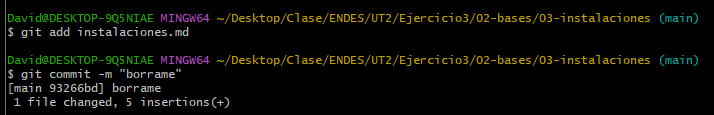
1. **Actualizar mensajes de commits y deshacer commits**
   1. **Actualizar el mensaje de un commit**

Para cambiar el mensaje del último commit realizado usaremos el siguiente comando:



* 1. **Borrar un commit**

Modificamos el archivo instalaciones.md (escribimos cualquier cosa) y hacemos commit:



Para borrar el último commit realizado usaremos el siguiente comando:

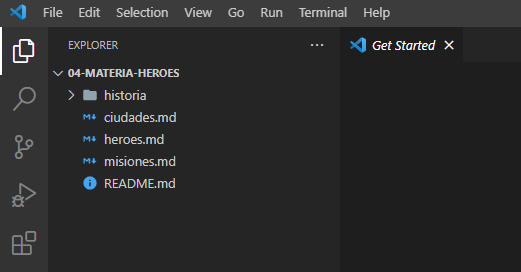


Esto eliminará el commit y nos dejará el repositorio como estaba antes de hacer dicho commit. Es decir si teníamos archivos modificados volverán a aparecer como modificados. El mantener los cambios se debe al parámetro --soft.

1. **Preparando un repositorio para viajes en el tiempo**

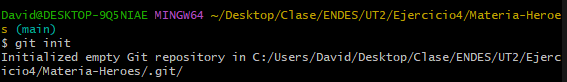
Descomprimimos el archivo Materia-Heroes y colocamos la carpeta Materia-Heroes en la carpeta donde estemos creando nuestros repositorios.

Renombramos la carpeta como: 04-Materia-Heroes y arrastramos la carpeta a visual estudio code. Deberíamos tener algo así:



Vamos a ir ejecutando los siguientes comandos:

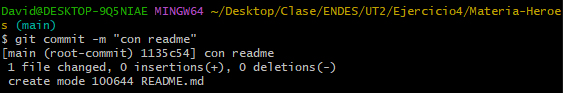
Inicializamos el repositorio:



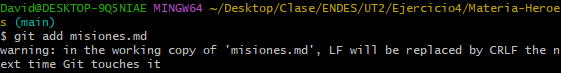
Añadimos el archivo README.md



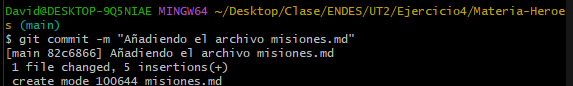
Hacemos el commit de este archivo:



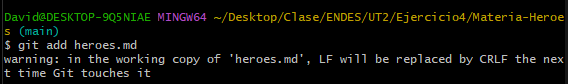
Añadimos el archivo misiones.md



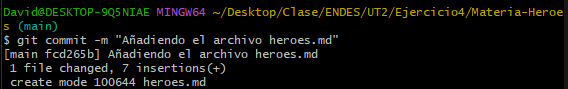
Hacemos el commit de este archivo:



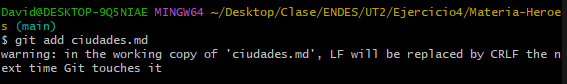
Añadimos el archivo heroes.md



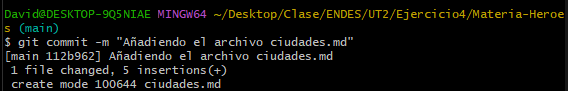
Hacemos el commit de este archivo:



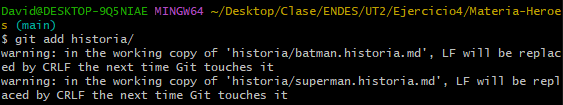
Añadimos el archivo ciudades.md



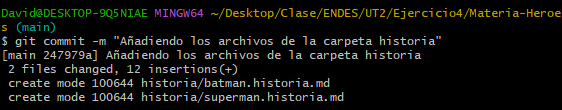
Hacemos el commit de este archivo:



Agregamos la carpeta historia

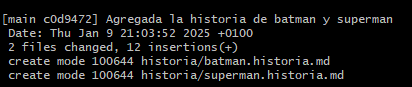


Hacemos el commit de los archivos agregados:

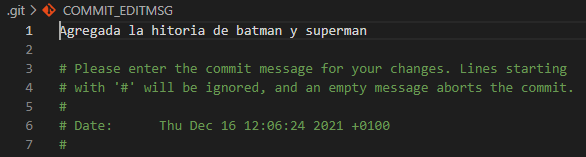


Cambiamos el texto del último commit:

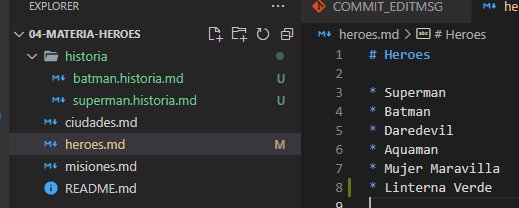




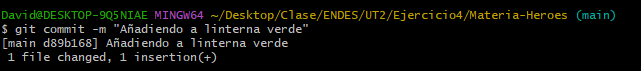
Al ejecutar el comando sin poner al final –m “nuevo texto para el commit” se abrirá el editor y podremos cambiar en él el texto del commit:



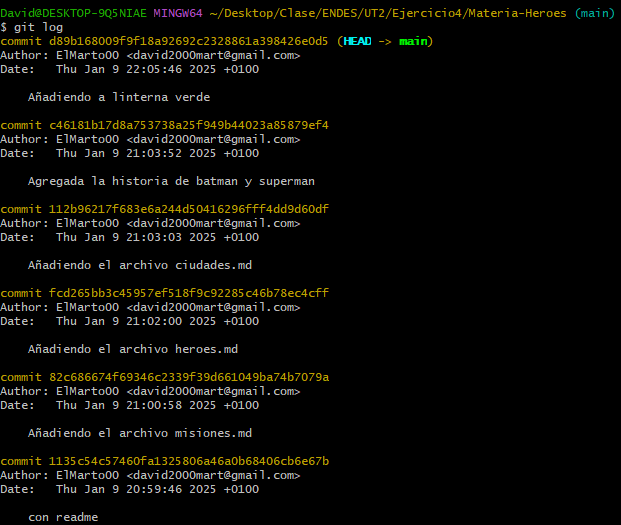
Editamos el archivo héroes.md y agregamos al héroe Linterna Verde:



Hacemos un nuevo commit:



Ejecutamos el comando git lg para ver un listado de los commits que hemos realizado: (Pega una captura)

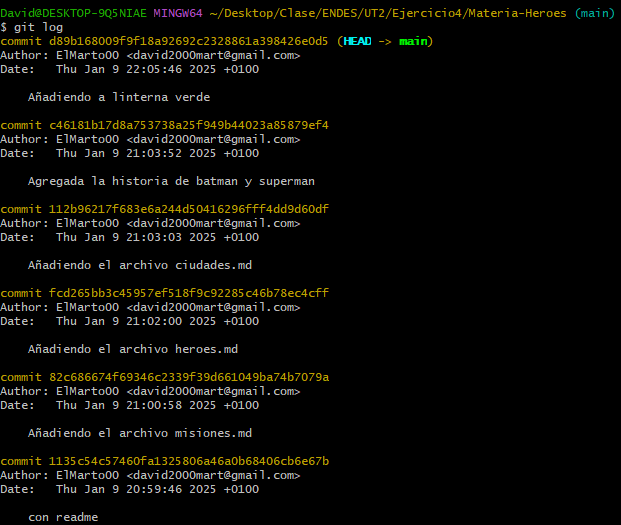


1. **Viajes en el tiempo, resets y reflogs**

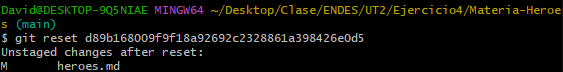
Vamos a añadir el héroe Robin en el archivo heroes.md:

En vez de hacer un nuevo commit para añadir este cambio queremos que haya un solo commit con la agragación de Robin y Linterna Verde.

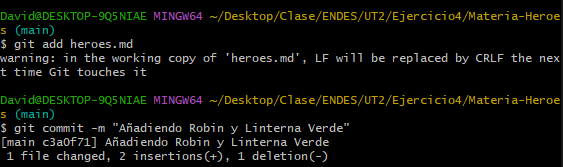
Hacemos un git lg: Copia una captura e indica el código hash del commit al que queremos volver



Vamos a regresar a un commit anterior usando como referencia el código hash del commit al que queremos volver:



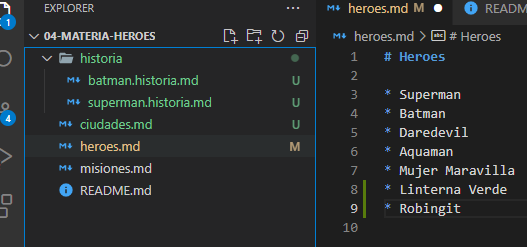
Hacemos un nuevo commit:



Ahora decidimos que los últimos cambios realizados no están bien y que queremos volver al punto en el que agregamos el archivo de heroes. Y además queremos que los archivos de los que se hubiera hecho commit se queden fuera del stage si es que los habíamos subido. Para ello hacemos un git reset --mixed:



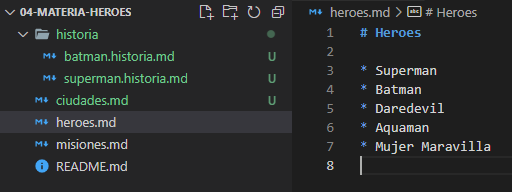
Si no ponemos una opción del modo de reset (--soft --mixed etc) el tipo de reset que se hace es el --mixed ya que es el modo por defecto.



Vemos que los archivos han pasado a estar con la U de untracked es decir sin seguimiento pero que los últimos cambios realizados en heroes.md se han mantenido.

Si queremos hacer un reset que borre los cambios usaremos la opción --hard:



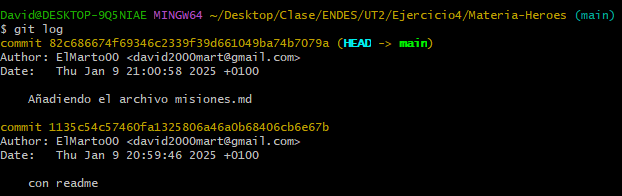


Vemos que el archivo heroes.md ha vuelto al estado del commit al que hemos regresado.

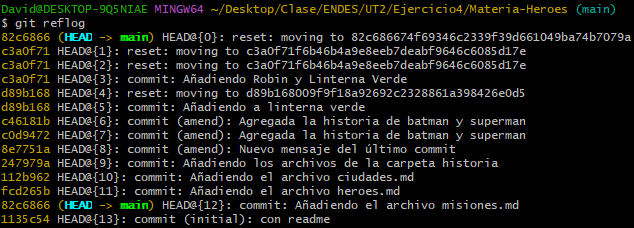
Hacemos un reset hard al commit donde agregamos las misiones:



Si hacemos un nuevo git lg: (copia captura)



Vemos que sólo nos quedan dos commits. Si ahora nos damos cuenta de que estaba todo bien y que queremos volver a un commit de los que borramos, todavía es posible ya que git guarda un historial:

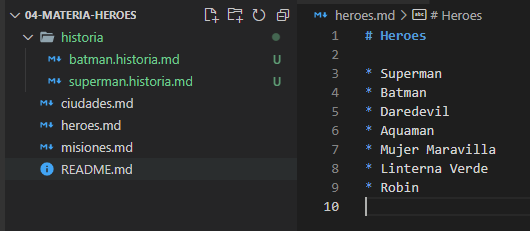


Copia captura

Como podemos ver en este historial se guardan tanto los commit como los reset que se fueron realizando. Esto nos permite ver los hash de los commit borrados y si por ejemplo queremos volver al commit en el que agregamos a los héroes Linterna Verde y Robin no tenemos más que ejecutar el siguiente comando:



Vemos que todo vuelve a ese punto:

****