Práctica GIT

Contenido

1. Iniciando el repositorio.

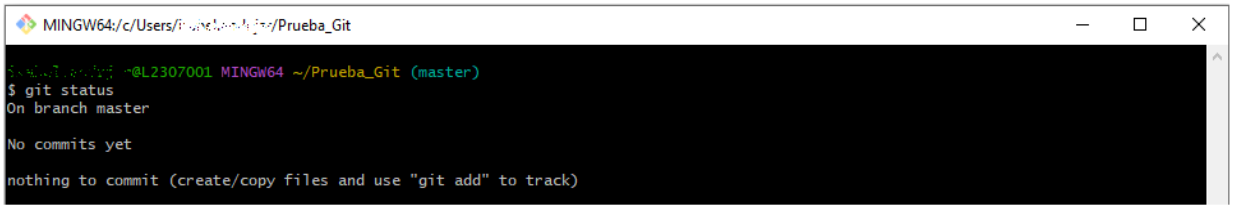
Para ir familiarizándonos con los comandos que podemos utilizar vamos a realizar un ejercicio práctico.

> ⚠ El ejercicio lo vamos a realizar con GIT Bash, la interfaz de línea de comandos de Git, ya que a pesar de que hay aplicaciones de escritorio gráficas que nos pueden ayudar, lo más importante es saber manejarse bien con los comandos.

En primer lugar, abriremos el terminal de Git (**Git Bash**) y creamos un directorio de prueba en el que vamos a trabajar. Una vez creado accedemos a él y ejecutamos el comando '**`git init`**'. Con este comando creamos un repositorio local en el directorio de trabajo.

La terminal nos informa de que hemos iniciado el repositorio y que ha creado en el 'fichero oculto' '.git'. Es importante mantener esta carpeta segura ya que si se pierde entonces perderemos el repositorio.

Una vez creado el repositorio, la carpeta en la que estamos sería nuestro ***working directory***. Para ver el estado en el que se encuentra usamos el comando '**`git status`**':



Este comando nos indica la rama en la que estamos trabajando y que no hemos realizado ningún ***commit*** hasta la fecha. Esto es lógico ya que acabamos de crear el repositorio y aún no hemos trabajado en él.

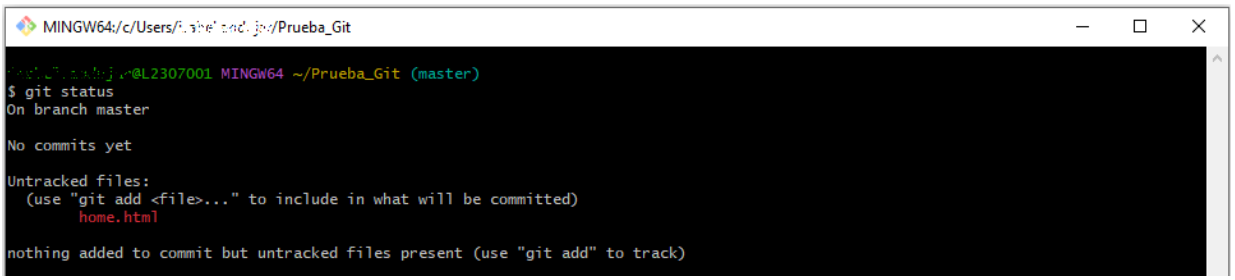
Un ***commit*** representa un punto en la historia del proyecto donde hemos guardado cambios específicos. En otras palabras, es algo así como una captura de tu proyecto suele ir acompañado de una breve descripción que aclara qué nuevas modificaciones se han hecho. Este comando crea una nueva versión del proyecto en tu zona de trabajo local (***staging area***).

Cada ***commit*** tiene un identificador único que nos permite realizar un seguimiento de la evolución del código a lo largo del tiempo y tener un registro de los cambios realizados y por lo tanto, podríamos incluso revertir cambios.

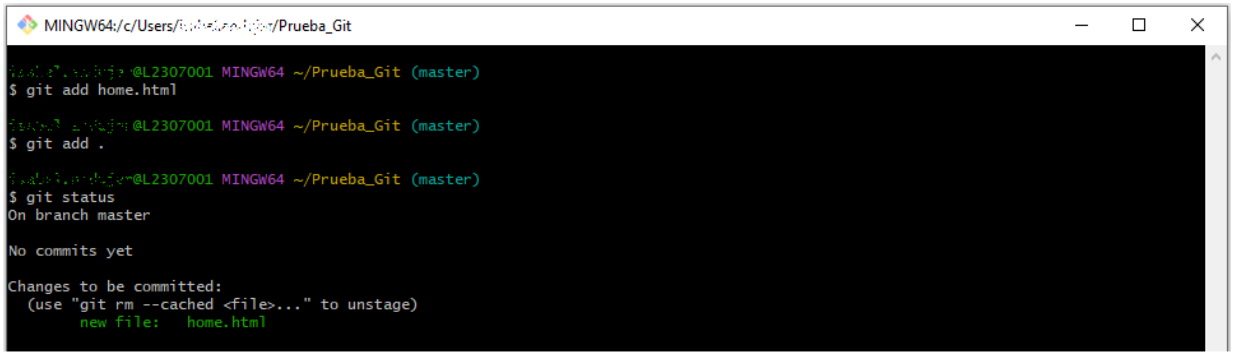
2. Trabajando en el repositorio.

Para comenzar a trabajar en el repositorio vamos a copiar el archivo '***home.html***' en el directorio que creamos en el paso anterior. El archivo contiene la siguiente información:

Una vez realizado el cambio, volvemos a comprobar el estado de nuestro repositorio ('**`git status`**'):



Vemos que Git nos avisa de que el archivo que acabamos de agregar en el directorio no está en seguimiento. Para añadir los archivos al seguimiento utilizamos el comando '**`git add`**':



El comando ***git add*** puede ser utilizado principalmente de dos formas:

1. **'git add nombre_archivo'**: Para agregar un archivo especifico.
2. **'git add .'**: Para agregar todos los archivos modificados y nuevos en el directorio actual.

Utilizando este comando estaríamos 'colocando' los archivos en el área de preparación o ***staging area***. Los archivos en esta área están listos para ser incluidos en el próximo commit. En otras palabras, hemos indicado a Git que deseamos incluir esos cambios específicos en nuestro historial de versiones.

Es importante tener en cuenta que los archivos en el ***staging area*** no se han confirmado completamente en el repositorio de Git, será necesario que realicemos un ***commit*** para que esos cambios se guarden permanentemente en el historial de versiones. Para ello usamos el comando '**git commit**'

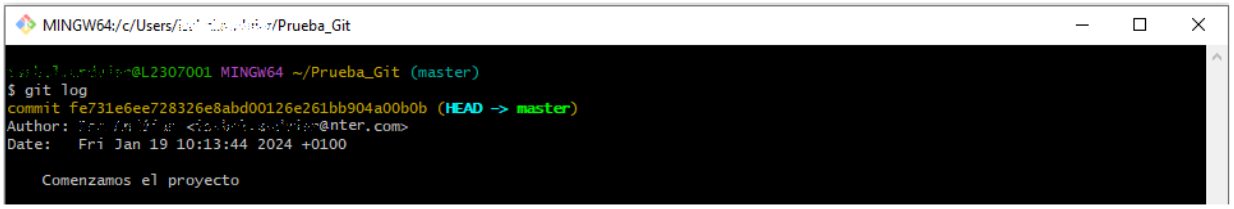
Este comando coge los archivos que hemos agregado previamente con '**git add**' al área de preparación y los guarda como un nuevo ***commit*** en el historial de versiones de Git. El mensaje del commit proporciona una descripción breve de los cambios realizados en ese commit en particular.

En comando **'git commit -m "Mensaje del commit"'** el parámetro *'-m "Mensaje del commit"'* proporciona una breve descripción de los cambios realizados en ese ***commit*** en particular.

Nota: Si no usamos el parámetro '-m', cuando ejecutemos el comando se nos abrirá un editor de texto para que veamos el cambio que vamos a realizar y podamos añadir la descripción.

¡Con esto hecho ya tenemos nuestro primer commit!

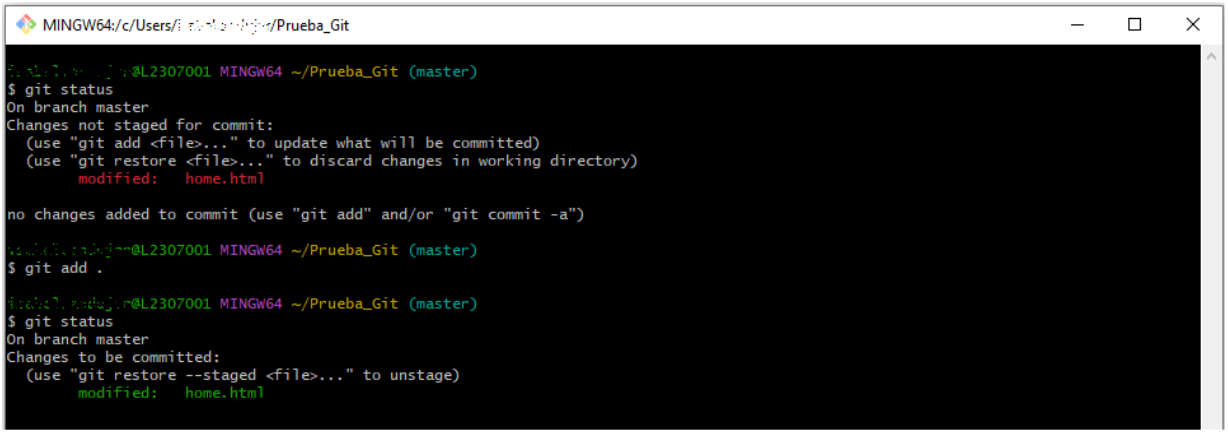
Para revisar el ***commit*** que acabamos de hacer usaremos el comando con '**`git log`**'. Este comando nos permite ver el historial de commits en el repositorio de Git. Al ejecutarlo, veremos una lista de commits junto con información relevante, como el autor, la fecha, y el mensaje asociado al commit.



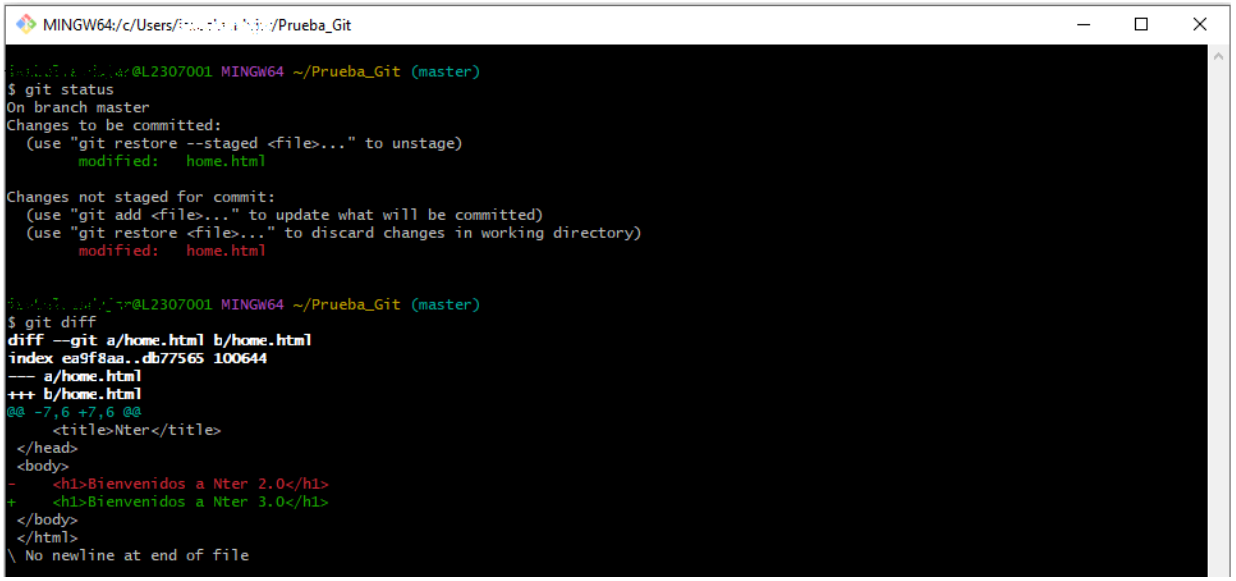
Vemos que nuestro commit tiene un código asociado a él, '`fe731e6ee728326e8abd…'`. Este código nos servirá para poder identificarlo y, como veremos más adelante, volver a ese punto si fuese necesario.

Si no queremos ver tanta información acerca de los commits, el parámetro **'--oneline'** hará que se nos muestre la información en una sola línea.

Ahora que hemos realizado nuestro primer commit, vamos a empezar a hacer cambios en el repositorio y ver qué ocurre. Para ello, en primer lugar, debemos hacer alguna modificación en el archivo. Esto lo podemos hacer abriendo el archivo en cualquier editor de texto plano y por ejemplo, escribiendo 'Bienvenidos a Nter 2.0' y una vez hecho, seguimos los siguientes pasos:



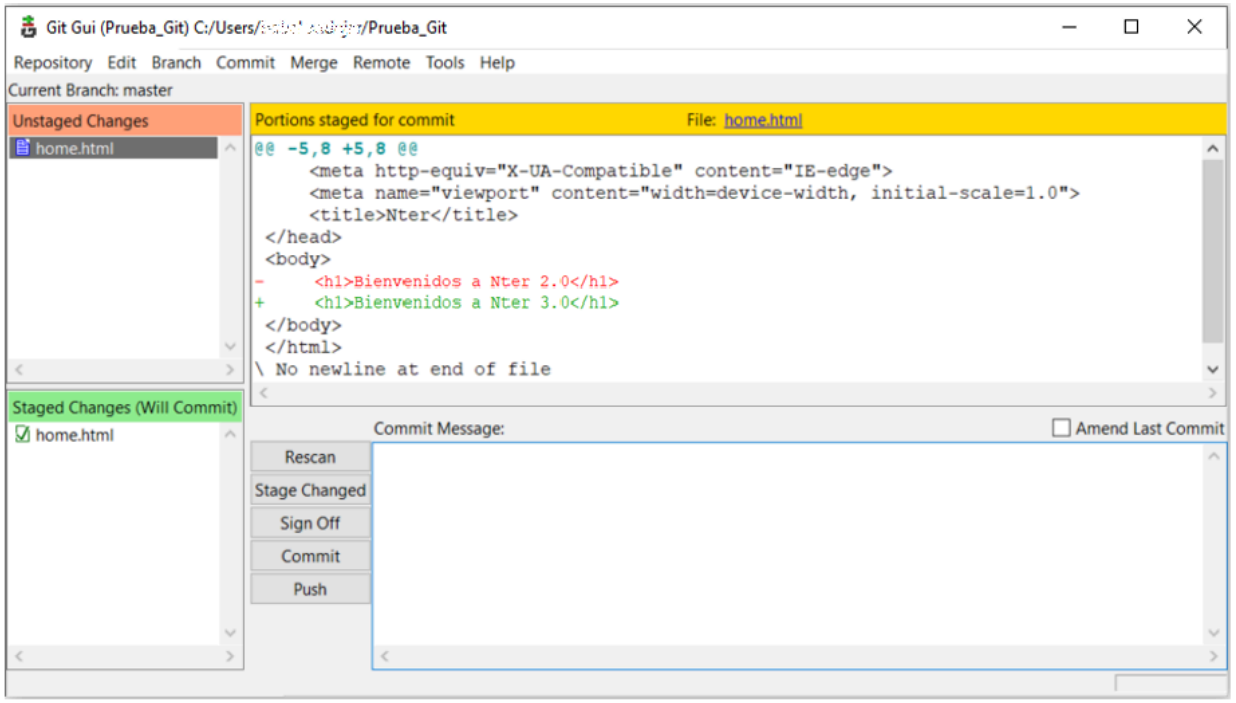
A continuación, volvemos a modificar el archivo y hacemos lo siguiente:



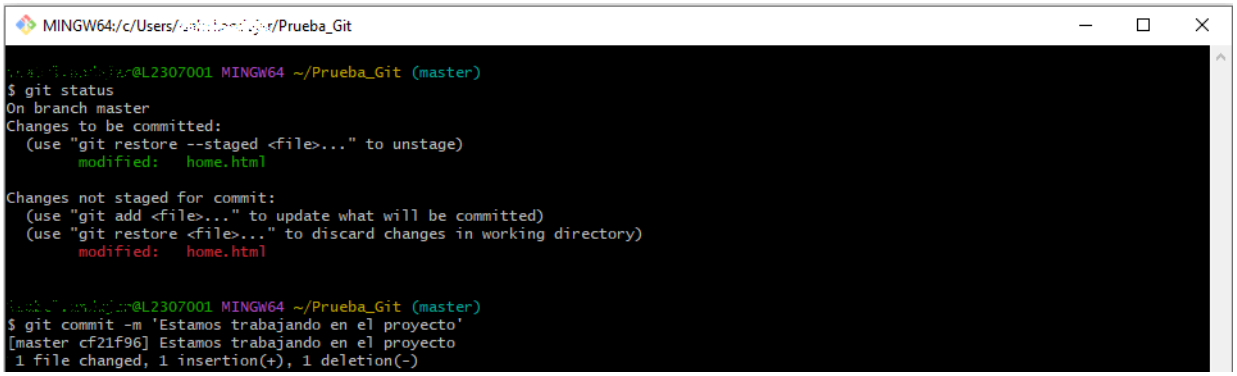
Como vemos tenemos cambios guardados en el ***staging area*** y, posteriormente se han vuelto a hacer cambios sobre el mismo fichero.

Para ver las modificaciones que se han hecho en fichero con respecto a lo que tenemos en el ***staging area*** podemos utilizar el comando '**`git diff`**'. Este comando nos muestra en rojo la línea modificada que está guardada en el ***staging area*** y en verde la línea de la última modificación que se ha realizado.

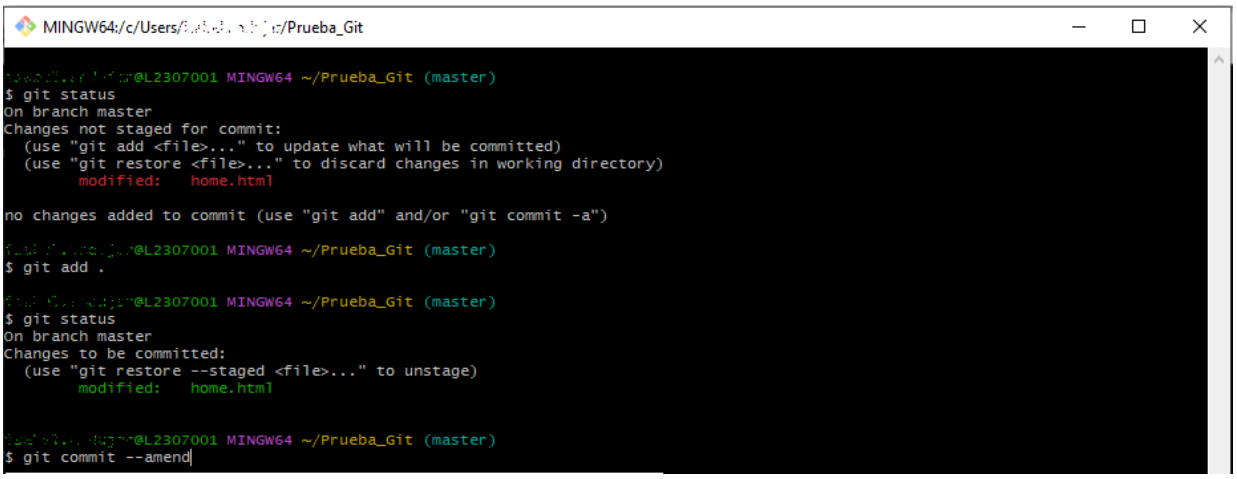
Otra opción para ver estos cambios es usar **'Git Gui':**



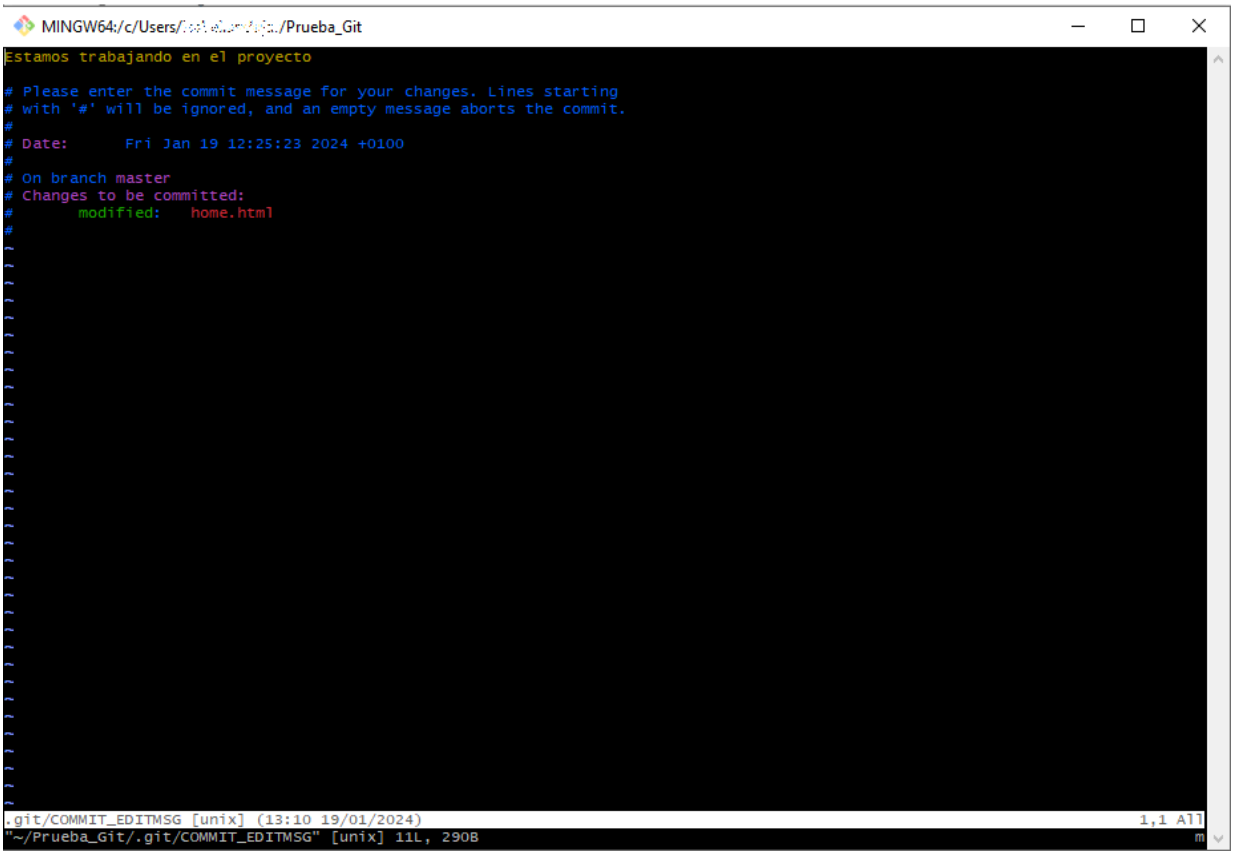
Si ahora hacemos un ***commit***, debemos tener en cuenta que solo se almacenarán los cambios que estaban en el ***staging area*** y, por lo tanto, si queremos almacenar todos los cambios debemos añadir las últimas modificaciones.



No obstante, si hacemos ese ***commit***, tenemos la opción de agregar esos cambios al commit anterior, para ellos usamos el comando '**`git commit --amend`**' :

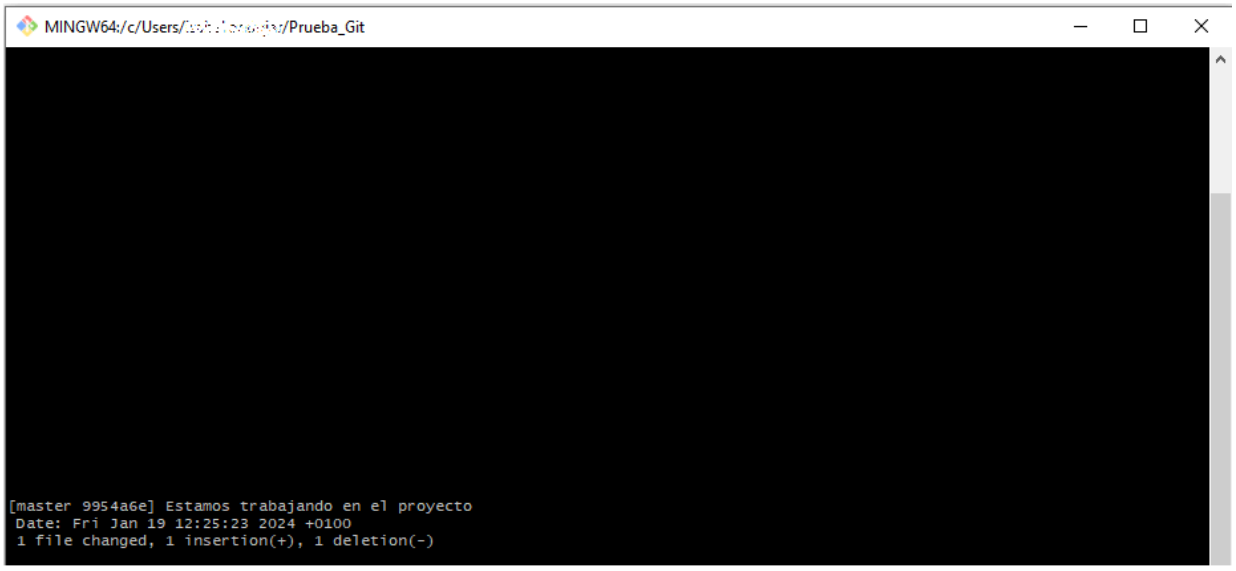


Una vez ejecutemos el git commit --amend se nos abrirá 'Vim' un editor de código en el que GIT nos va a indicar el mensaje descriptivo del commit al que vamos a agregar estos cambios de última hora, junto con los ficheros involucrados, la fecha, etc. Será algo parecido a lo siguiente:



Ahora para aceptar los cambios debemos:

1. Presiona la tecla Esc para asegurarte de estar en modo de comandos.
2. Escribe '**:wq**' y presiona Enter. (Esto guarda los cambios y cierra Vim)



Si se te ha abierto en VIM y no sabes salir, tienes que escribir la combinación de teclas '**:qa!**'

¡Listo! Con esto ya hemos terminado la primera práctica.