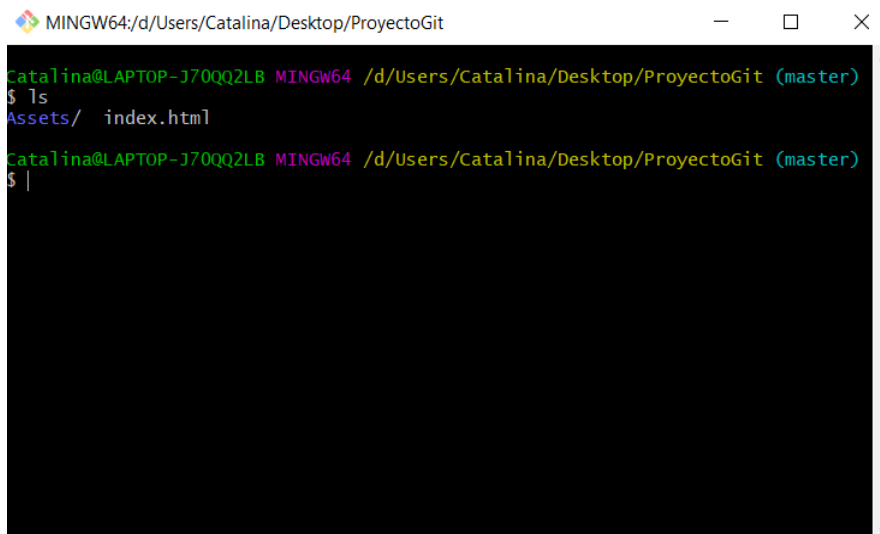


## HINTS

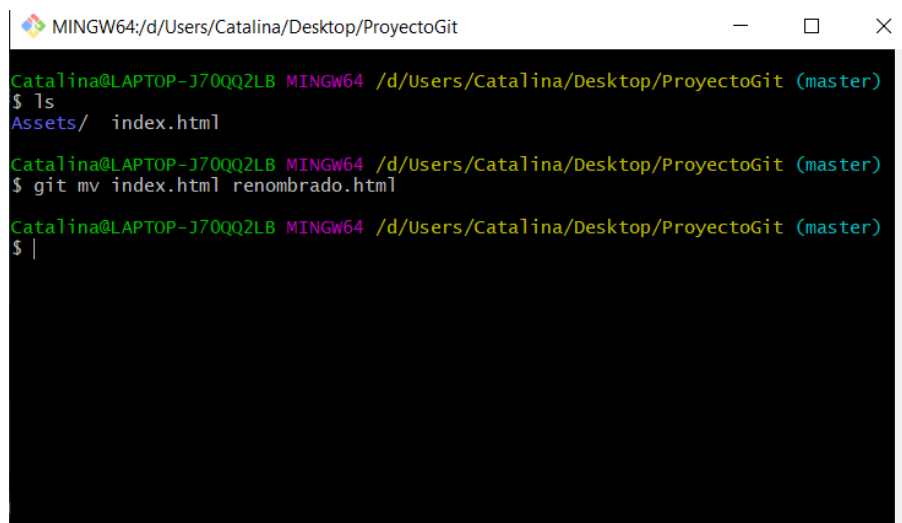
### GIT MV

El comando **git mv** nos permite renombrar un archivo. Para eso, evaluemos, con el comando **ls**, los archivos con los que contamos.



```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ ls
Assets/  index.html
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ |
```

Utilicemos ahora el comando **git mv** seguido del nombre actual y el futuro nombre.



```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ ls
Assets/  index.html
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git mv index.html renombrado.html
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ |
```

Listemos nuevamente los archivos para corroborar que el cambio de nombre de haya realizado.

```

MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ ls
Assets/ index.html
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git mv index.html renombrado.html
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ ls
Assets/ renombrado.html
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$

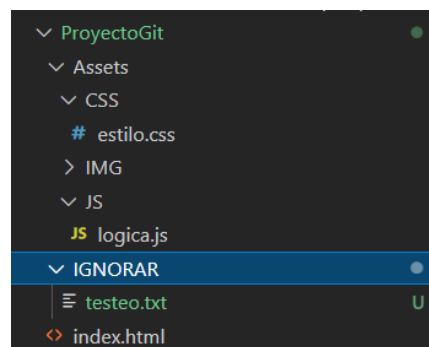
```

De esta forma podemos utilizar la consola de **git bash** para renombrar.

## GIT IGNORE

Dentro de las posibilidades de trabajo, existirán archivos que queramos ignorar porque los utilizaremos para prueba, para registro o cualquier posibilidad que no queremos que se pueda incluir dentro de los **commit**. Para hacer eso **GIT** nos proporciona **.gitignore**, que ignorará el archivo que se le indique.

Creemos una carpeta llamada "IGNORAR" y en ella un archivo **testeo.txt**.



Si hacemos un **git status** en la consola, veremos la nueva carpeta.

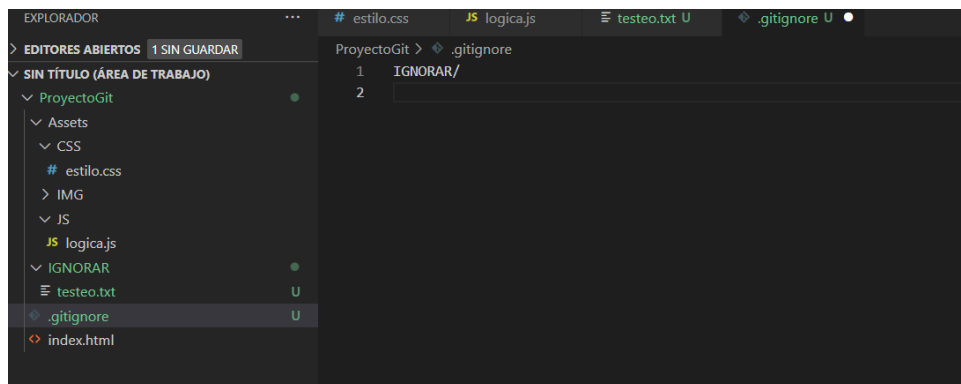
```

MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    IGNORAR/

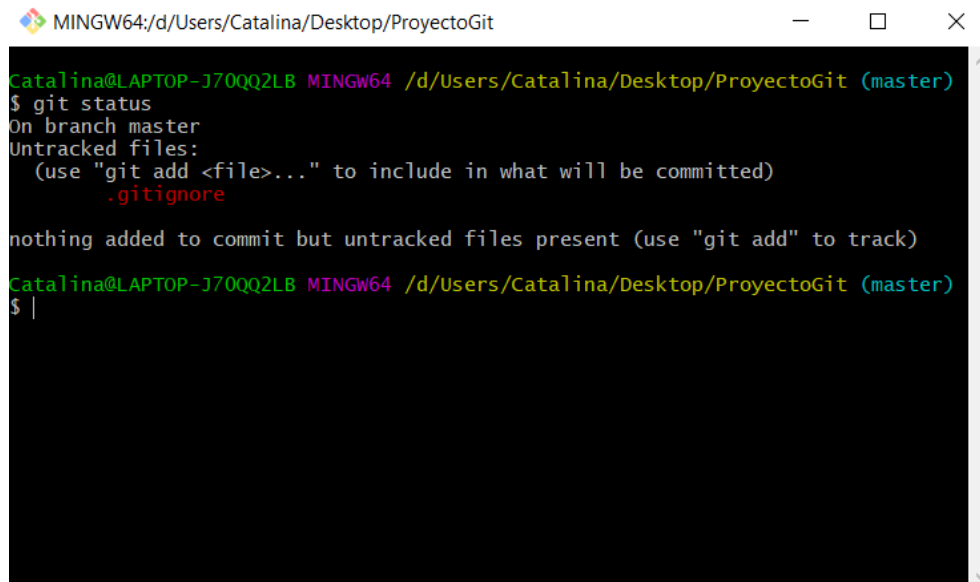
nothing added to commit but untracked files present (use "git add" to track)
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ |

```

Esto es lo que no deseamos que ocurra. Para evitar que esto ocurra, en la raíz del proyecto crearemos un archivo de nombre **.gitignore** y dentro de él, el nombre de la carpeta que queremos ignorar, seguido de un slash y los guardamos.



Nuevamente hacemos un **git status** y podemos observar que ahora no nos muestra el fichero **IGNORAR**, pero si el archivo **.gitignore**.



```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ |
```

Simplemente debemos hacer un **commit** como ya lo sabemos hacer y ya nuestro proyecto ignorará los cambios que se realicen en los archivos indicados, en este caso, cualquier archivo dentro de la carpeta **IGNORAR**.

## TRABAJAR CON RAMAS

Vimos como crear ramas, pero estas también se pueden eliminar o fusionar. Para eliminar una rama usaremos el comando:

```
Git branch -d nombre_rama
```

Este comando servirá siempre que la rama no tenga trabajados sin funcionar, ya que, de ser así, nos enviará un error.

La eliminación se puede forzar incluso sin hacer la fusión utilizando el comando:

```
Git branch -D nombre_rama
```

Ahora, si deseamos unir o fusionar ramas tendremos los comandos:

```
Git Merche  
git rebase.
```

Que nos permiten unir las líneas independientes de desarrollo en una sola rama. La diferencia entre ambas es que **git Merche** crea un nuevo commit que une a ambas ramas, sin eliminarlas. Es decir, se mantiene su historial, mientras que **git rebase** no realiza este nuevo commit, si no que reescribe la rama principal integrando los commit de la secundaria.

## GIT TAGS

El concepto de etiquetado en Git hace referencia a etiquetas que marcan un punto concreto en la historia. Podríamos decir que es como una rama alternativa que no cambia, ya que no puede recibir más confirmaciones y se mantiene.

Podemos aprender más sobre [Git Tags](#) en el enlace externo compartido.

## COMANDO GIT STASH

Este comando almacena o guarda temporalmente los cambios que se hayan efectuado en el código en el cual estemos trabajando para que se pueda trabajar en otra cosa y más tarde retornar a este trabajo y aplicar los cambios.

Esto puede parecer poco productivo, pero resulta practico si se tiene que cambiar rápidamente de contexto y aun no tenemos el código listo para confirmar los cambios.