Práctica 03 – Primeros comandos

Contenido

1 Cambiar el nombre de la rama Master a Main

En general, una **rama de desarrollo** ("Git Branch") es una bifurcación del estado del código que crea un nuevo camino para la evolución del mismo.

El comando git branch nos indica en que rama estamos trabajando:

```
git branch
```

(Pega aquí una captura de la ejecución del comando)

1.1 Cambiar el nombre de una rama

Para cambiar el nombre de una rama usamos el siguiente comando:

```
git branch -m master main
```

El -m indica que se va a cambiar de nombre a una rama.

A continuación, ponemos el nombre de la rama a renombrar (en este caso master).

Finalmente indicamos el nuevo nombre (en este caso main).

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT> git init
Reinitialized existing Git repository in C:/NIEVES/DAW/ENDES/Práctica GIT/.git/
PS C:\NIEVES\DAW\ENDES\Práctica GIT> git branch
* main
PS C:\NIEVES\DAW\ENDES\Práctica GIT> git branch -m master main
fatal: a branch named 'main' already exists
PS C:\NIEVES\DAW\ENDES\Práctica GIT> git branch -m main master
PS C:\NIEVES\DAW\ENDES\Práctica GIT> git branch
* master
PS C:\NIEVES\DAW\ENDES\Práctica GIT>
```

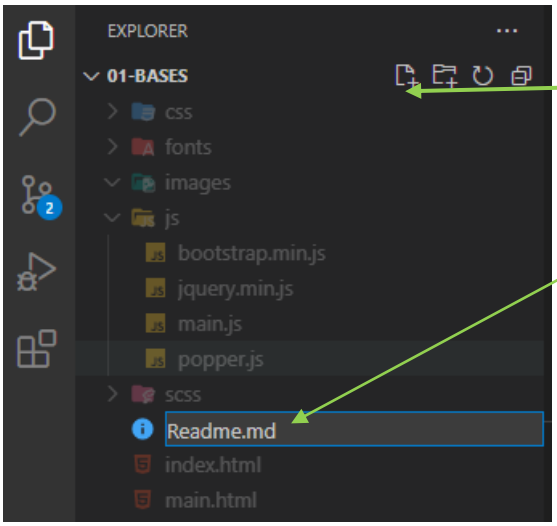
1.2 Configurar que por defecto la rama principal de un nuevo proyecto se llame main

```
git config --global init.defaultBranch main
```

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT> git config --global init.defaultBranch main
PS C:\NIEVES\DAW\ENDES\Práctica GIT>
```

2 Archivo Readme.md y comando log

Vamos a crear un archivo nuevo en nuestro proyecto, al que llamaremos Readme.md:

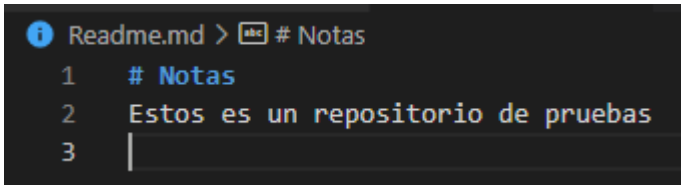


Pulsamos en el botón de nuevo archivo:

Y escribimos el nombre en el recuadro que aparece:

Un archivo README contiene información acerca de otros archivos en un directorio. Es una forma de documentación de software. En Github es habitual agregar un archivo README a un repositorio para comunicar información importante sobre el proyecto.

Escribimos algo de contenido en el archivo:



Le damos seguimiento al archivo:

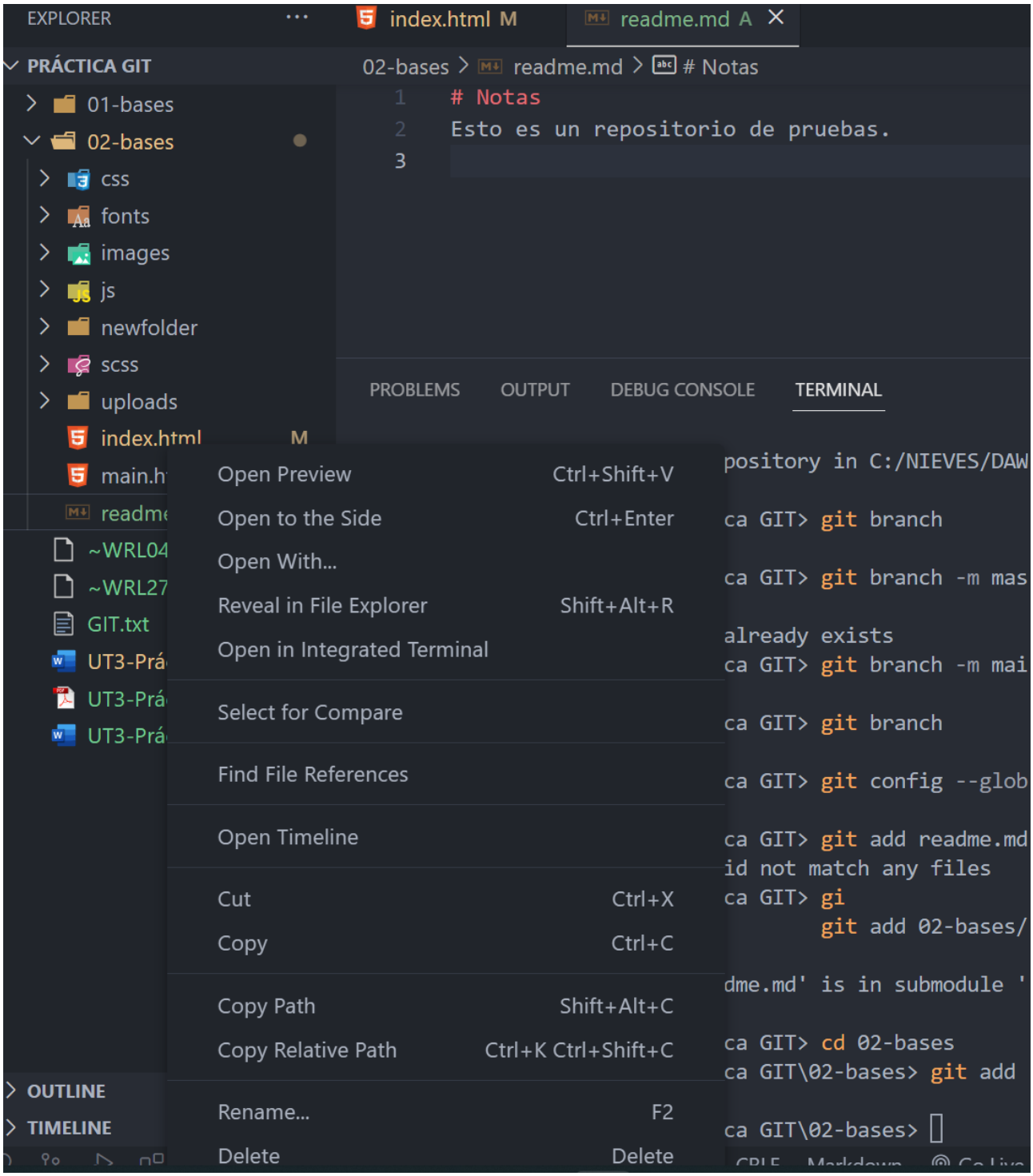
```
git add readme.md
```

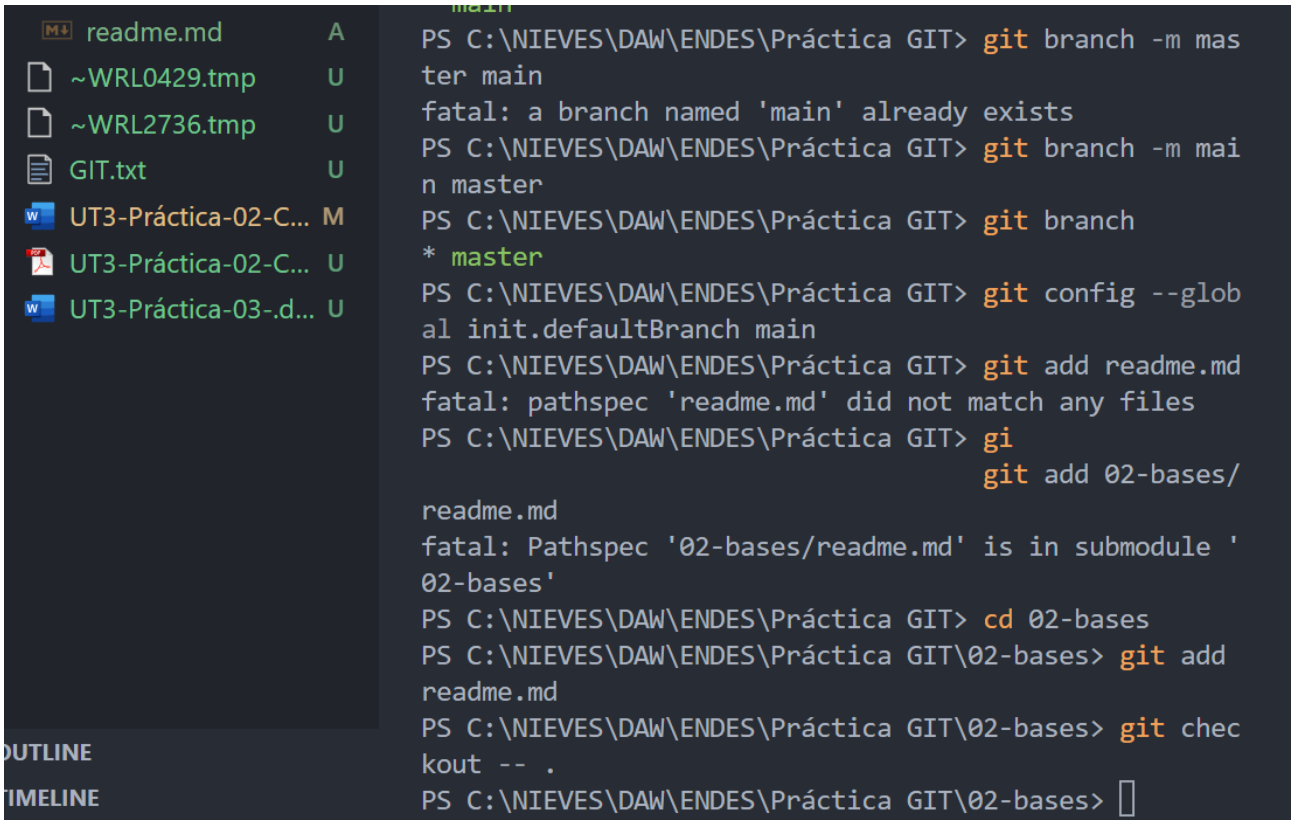
Hacemos commit:

```
git commit -add "readme modificado"
```

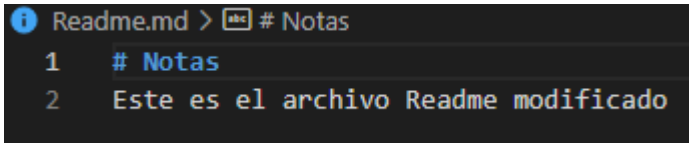
Borramos el archivo desde el visual estudio y como hemos hecho commit previamente podemos recuperarlo con:

git checkout -- .





Modificamos el archivo Readme:



Hacemos un segundo commit con una versión nueva del comando checkout:

```
git checkout -- .
```

Este comando con –a sólo funciona si ya le estamos dando seguimiento al archivo, pero si estuviera marcado con la U de “untracked” es decir sin seguimiento no funcionaria.

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git checkout -- .
```

2.1 Ver los commits realizados

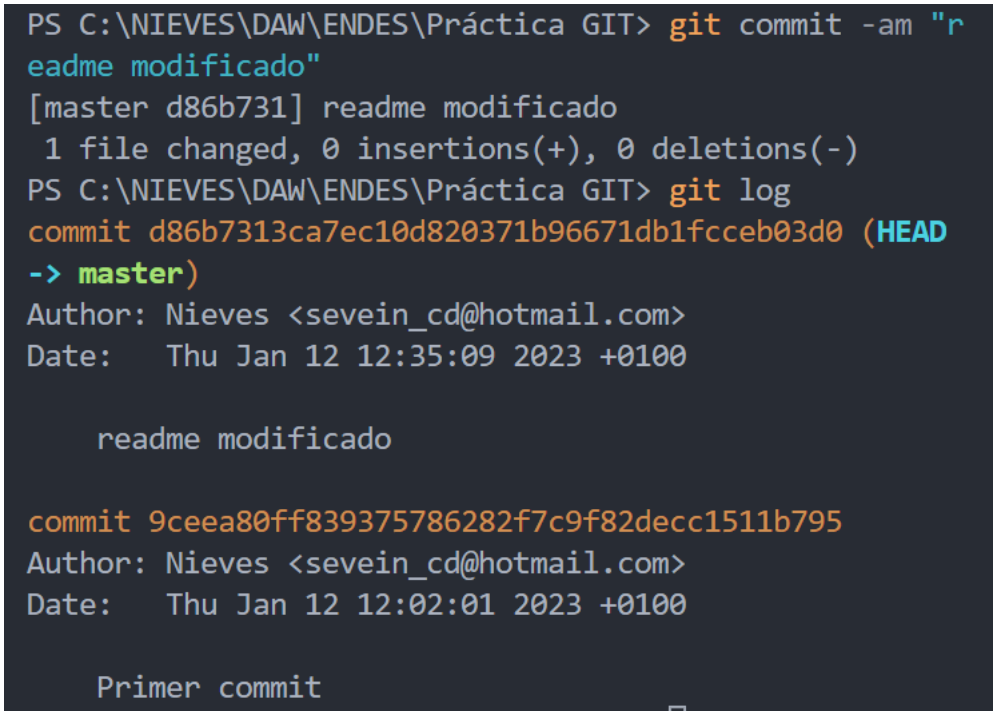
Vamos a ver los commits que tenemos hechos hasta el momento, para ello vamos a utilizar el siguiente comando:

```
git log
```

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git log
```

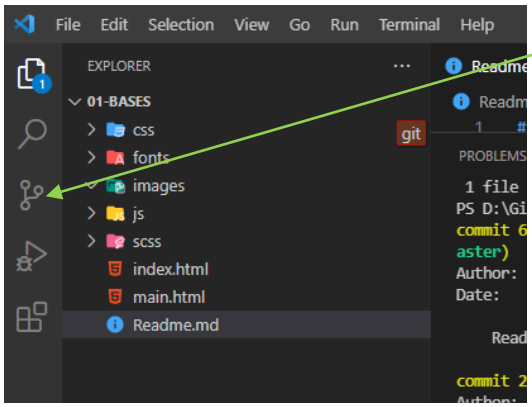
Identifica los siguientes elementos en la captura y señálalos:

- Última versión del repositorio: la que aparece arriba del todo
- Identificador hash del commit: d86b731ca7ec10d8…
- Autor del commit: Author: Nieves
- Fecha y hora del commit: Thu Jan 12 12:35
- Texto identificativo del commit: readme modificado

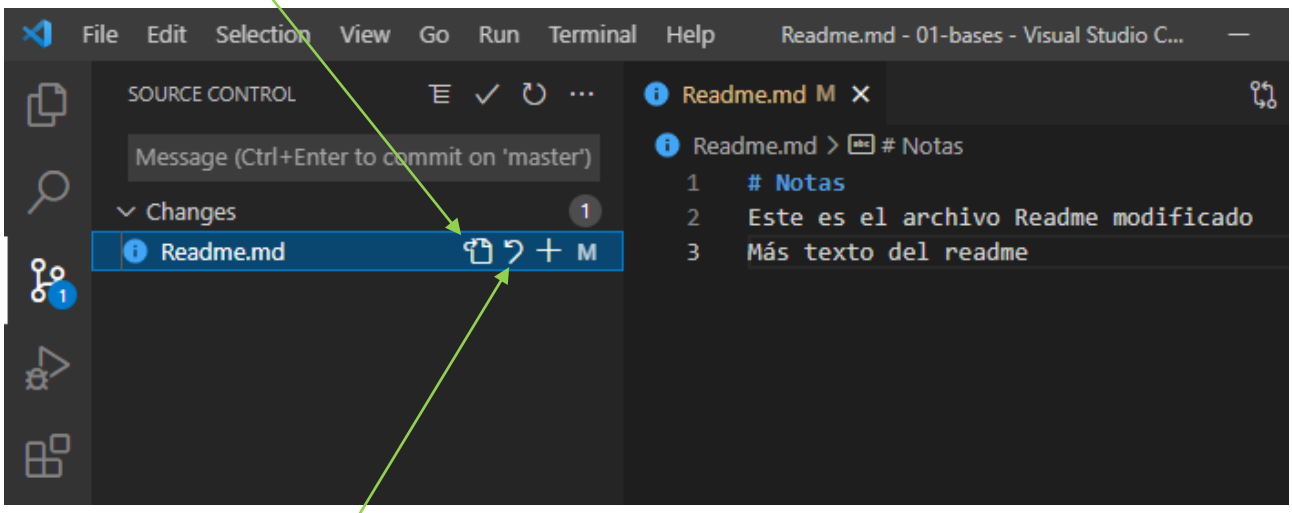


3 Adds y commits con visual studio code

En Visual studio code tenemos un apartado para realizar operaciones de git:

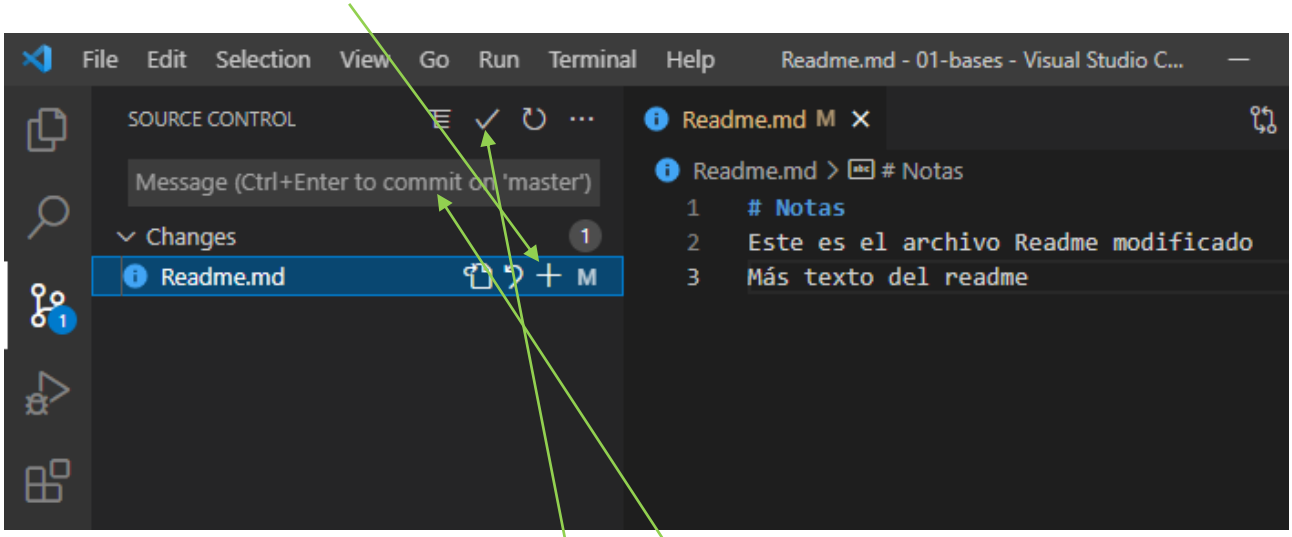


Con este icono abrimos el archivo:



Con este otro deshacemos los cambios realizados en el archivo:

Para añadir los cambios al escenario usamos este botón:



Y para hacer un commit escribimos el mensaje en este campo. Y a continuación utilizamos la combinación Ctrl+Enter o pulsamos en el botón de commit.

4 Diferentes formas de agregar archivos al escenario.

Cerramos en Visual Studio Code nuestro proyecto actual que era el de la carpeta 01-bases. Para ello vamos al menú File -> Close Folder o pulsamos la combinación Ctrl + K y luego F.

Descomprimimos el archivo 02-bases.zip y copiamos la carpeta 02-bases a la carpeta donde habíamos copiado la carpeta 01-bases.

Arrastramos esta carpeta a visual studio code como hicimos con 01-bases

Nos situamos en la carpeta 02-bases e inicializamos el repositorio:

```
git init
```

Vamos a ver cómo hacer commits de grupos de archivos en vez de commit de todos los archivos como hicimos en apartados anteriores.

4.1 Añadir archivos sueltos

```
git add index.html main.html
```

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git init
Reinitialized existing Git repository in C:/NIEVES/DAW/ENDES/Práctica GIT/02
-bases/.git/
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git add index.html main.html
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   readme.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.md

PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git add *.html
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   readme.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
```

Podemos añadir archivos sueltos simplemente escribiendo sus nombres separados por espacios.

4.2 Añadir archivos usando el comodín *

Podemos añadir todos los archivos html de la siguiente forma:

```
git add *.html
```

A continuación hacemos commit de estos archivos

```
git commit -add "html añadido"
```

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git init
Reinitialized existing Git repository in C:/NIEVES/DAW/ENDES/Práctica GIT/02
-bases/.git/
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git add index.html main.html
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   readme.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.md

PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git add *.html
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   readme.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
```

Si ahora intentamos añadir todos los archivos .js

```
git add *.js
```

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT\01-bases> git add *.js
```

Esto no va a funcionar porque en el directorio actual no hay archivos .js , los archivos .js están en la carpeta js por lo que vamos a tener que indicarlo a la hora de ejecutar el comando:

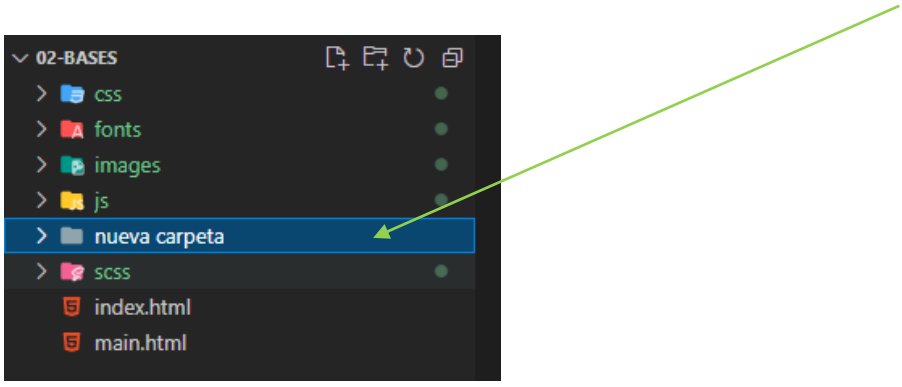
```
git add js/
```

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT\01-bases> git add js/
```

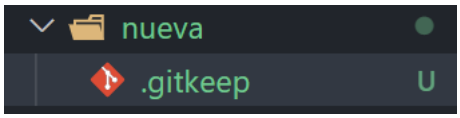
(**Nota**: Parece que en la versión actual si ponemos add *.js si que busca en las carpetas a partir de la carpeta actual y añade todos los archivos .js)

4.3 Carpetas vacías

Git no hace seguimiento a las carpetas vacías. Vamos a crear una carpeta nueva para verlo.

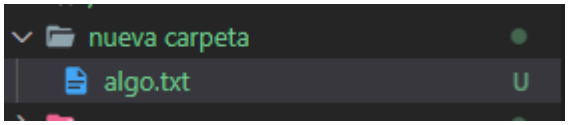


Si hacemos ahora git status:

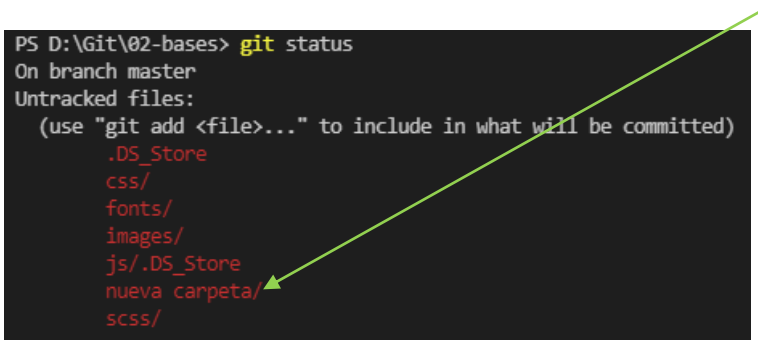


Vemos que git ni la muestra en la lista de archivos de los que no se hace seguimiento.

Si queremos que git la tenga en cuenta tenemos que crear un archivo dentro:



Y si ahora hacemos git status, vemos que ahora si aparece la nueva carpeta:

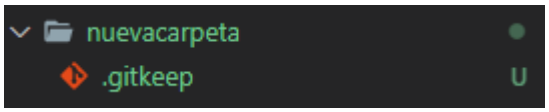


4.4 Archivo .gitkeep

Cuando queramos añadir al menos un archivo a una carpeta vacía para que git la tenga en cuenta, en vez de crear cualquier archivo existe un archivo con un nombre especial que podemos crear: el archivo **.gitkeep**

Este archivo está especialmente pensado para realizar esta función y ocupa un espacio muy pequeño.

(**Nota:** Renombramos "nueva carpeta" como "nuevacarpeta")



Vemos además que visual studio code le pone un icono específico al archivo con este nombre.

Añadimos este archivo con el siguiente comando:

```
git add nueva
```

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> gi
t status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unsta
ge)
        new file:   nueva/.gitkeep
```

En algunas versiones este comando puesto así podría fallar y podríamos usar esta otra opción:

```

```

(Pega aquí una captura de la ejecución del comando)

Hacemos commit:

```
git commit -m ".gitkeep agregado"
```

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git commit -m
".gitkeep agregado"
```

Si borramos la carpeta y restauramos con:

```
git checkout -- .
```

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> git checkout -- .
```

Vemos que se restaura tanto el archivo .gitkeep como la carpeta que lo contenía.

4.5 Añadir una carpeta y todo su contenido

Si queremos añadir por ejemplo todo el contenido de la carpeta css (tanto archivos como directorios que contiene) podemos hacerlo con el siguiente comando:

```
git add css\
```

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> gi
t add css\
```

Hacemos commit:

```
git commit -m "estilos agregados"
```

```
PS C:\NIEVES\DAW\ENDES\Práctica GIT\02-bases> gi
t commit -m "estilos agregados"
[main b25f1fb] estilos agregados
 2 files changed, 2 insertions(+)
 create mode 100644 nueva/.gitkeep
 create mode 100644 readme.md
```