

EXERCISES QUE TRABAJAREMOS EN LA CUE

- EXERCISE 1: CONOCIENDO GIT Y COMANDOS BÁSICOS DE LINUX.
- EXERCISE 2: COMANDOS BÁSICOS DE GIT.
- EXERCISE 3: GIT DESDE VISUAL STUDIO CODE.

EXERCISE 1: CONOCIENDO GIT Y COMANDOS BÁSICOS DE LINUX.

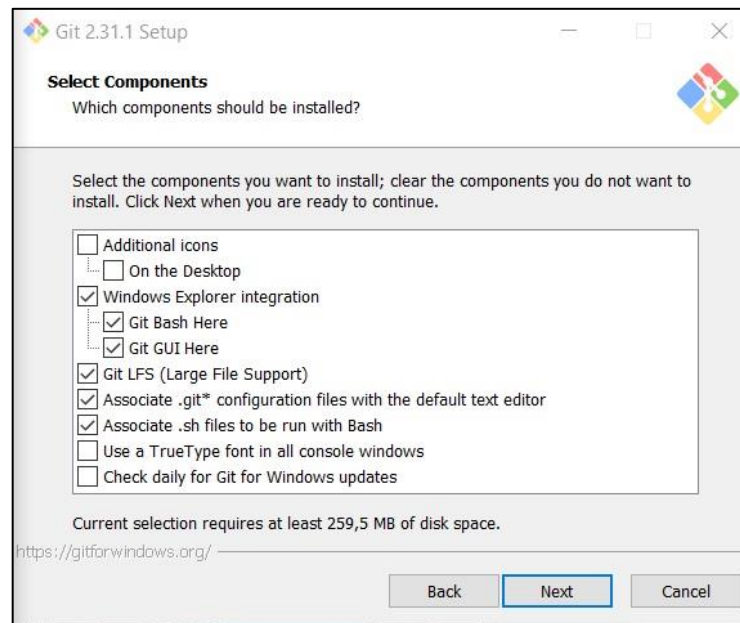
Hemos estado analizando la importancia de llevar un control de las versiones de nuestro software, pero aún no hemos entrado a conocer cómo podemos realizar este versionamiento y cómo utilizar **GIT** a nuestro favor.

Para comenzar instalaremos **GIT**. Debemos dirigirnos a la página oficial de [GIT](#) y descargaremos la versión que se encuentre disponible haciendo clic en el icono de descarga.



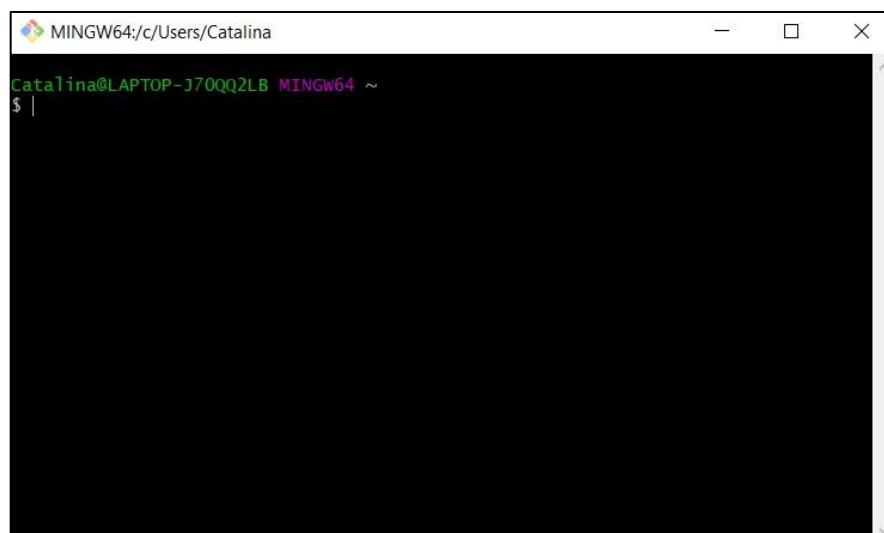
Se muestran varias versiones disponibles para descarga, pero inmediatamente se descarga la versión disponible.

Una vez que termine la descarga, vamos a instalar **GIT** procurando seleccionar que incorpore **GIT Bash**.



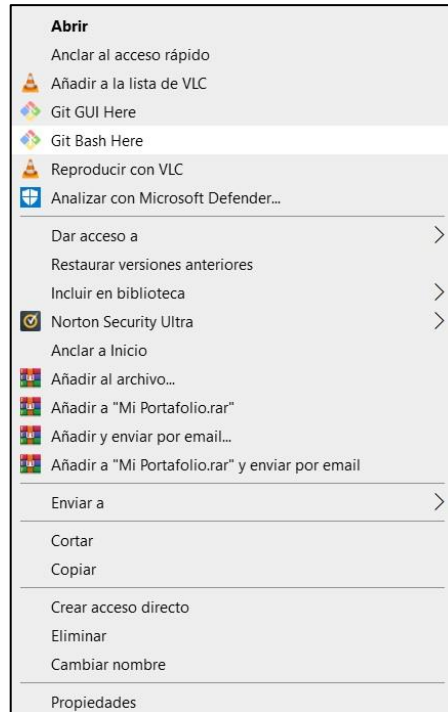
Cuando termina la instalación vamos a comenzar a conocer los comandos básicos que usaremos en la terminal (consola para interactuar con el computador) para trabajar con **GIT**.

Se abrirá una terminal que de **GIT Bash** que se verá de la siguiente forma:

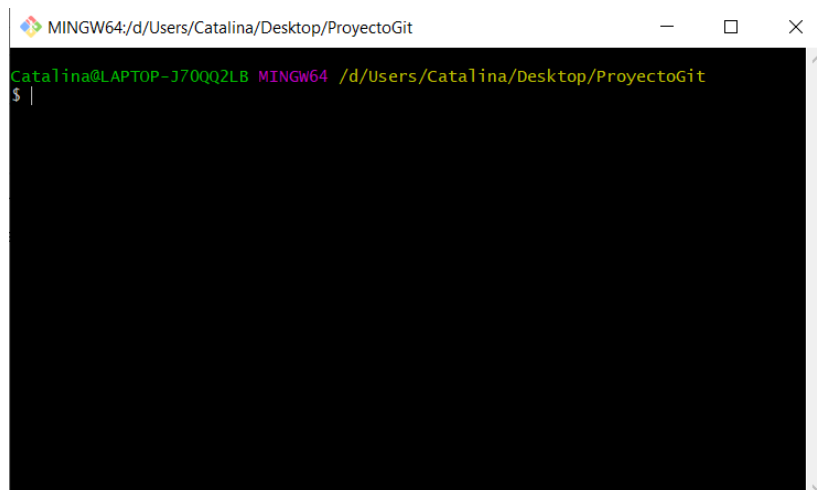


Para comenzar, cerraremos la ventana que se había abierto y crearemos una nueva carpeta a la cual llamaremos "ProyectoGit".

Sobre el proyecto, vamos a hacer clic secundario y seleccionaremos **Git Bash**.



Se abrirá nuestra terminal y en ella vamos a comenzar a trabajar:



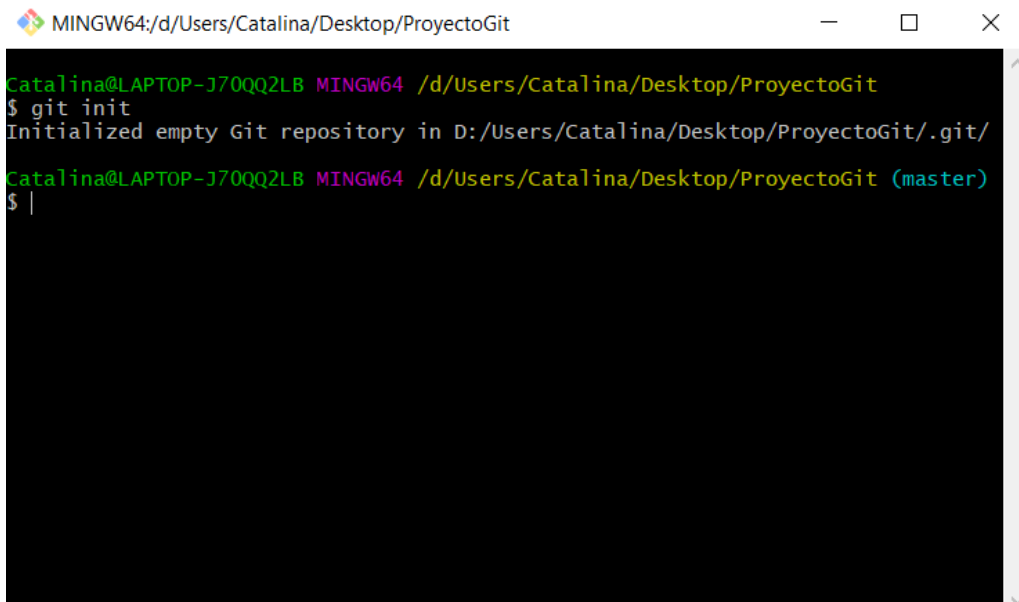
Comenzaremos conociendo comandos básicos de **Linux**, comandos de consola que se utilizan en conjunto con los comandos específicos de **GIT**.

Linux es un sistema operativo libre y gratuito. Nació de la combinación de varios proyectos de la mano de **Linus Torvalds** (el creador de **GIT**). Su desarrollo es uno de los mejores ejemplos del software libre, donde todo su código fuente puede ser utilizado, modificado y distribuido libremente.

COMENZANDO A DARLE FORMA A NUESTRO PROYECTO

Comenzaremos siguiendo nuestro proyecto, para eso usaremos el comando **git init**, el cual inicia un repositorio vacío, es decir, inicia un espacio en memoria en el cual se guardarán las versiones del proyecto.

Este comando es un comando básico de **GIT**, con el cual comenzaremos siempre el “seguimiento” de un archivo.



```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit
$ git init
Initialized empty Git repository in D:/Users/Catalina/Desktop/ProyectoGit/.git/
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ |
```

Ahora que hemos comenzado a seguir nuestro proyecto, crearemos el archivo **index.html**, para eso, utilizaremos el comando **touch**, indicando el nombre y la extensión del archivo. Este comando es parte de los comandos básicos de **Linux** con los que podemos trabajar en conjunto con **GIT**.

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit
$ git init
Initialized empty Git repository in D:/Users/Catalina/Desktop/ProyectoGit/.git/
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ touch index.html
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ |
```

Ahora que ya tenemos creado el archivo principal de nuestro proyecto desde la consola de **Git Bash**, continuaremos con la creación de un directorio, al cual llamaremos **Assets**.

Para cumplir esa misión usaremos **mkdir**.

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit
$ git init
Initialized empty Git repository in D:/Users/Catalina/Desktop/ProyectoGit/.git/
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ touch index.html
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ mkdir Assets
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ |
```

Si hemos realizado los pasos correctamente y revisamos nuestro archivo "ProyectoGit", encontraremos en él el archivo **index.html** y la carpeta **Assets**.

Para asegurarnos de eso, aprenderemos a listar los archivos que se encuentran en el proyecto que hemos creado, usando el comando `ls`.

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit
$ git init
Initialized empty Git repository in D:/Users/Catalina/Desktop/ProyectoGit/.git/
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ touch index.html
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ mkdir Assets
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ ls
Assets/ index.html
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$
```

Ahora podemos comenzar a navegar entre los archivos, usando el comando `cd`, ingresaremos al directorio **Assets** y en él crearemos las carpetas **IMG**, **CSS** y **JS**.

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit/Assets
$ mkdir Assets
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ ls
Assets/ index.html
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ cd Assets
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$ mkdir IMG
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$ mkdir CSS
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$ mkdir JS
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$
```

Acá es importante destacar que la consola nos muestra en que posición nos encontramos ubicados: hemos pasado de la ubicación en escritorio, **ProyectoGit** a la carpeta **Assets**, dentro del **ProyectoGit**.

Para terminar de crear nuestra estructura básica, vamos a ingresar al directorio **CSS** y crearemos el archivo **estilo.css**.

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit/Assets/CSS
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$ mkdir IMG
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$ mkdir CSS
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$ mkdir JS
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$ cd CSS
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets/CSS (master)
$ touch estilo.css
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets/CSS (master)
```

Antes de continuar, aprenderemos a limpiar la pantalla de la consola. Para eso escribiremos el comando **clear**.

Para dirigirnos ahora a la carpeta **Assets** nuevamente e ingresar al directorio **JS** a crear el archivo **lógica.js**, usaremos el comando **cd ..** (dos puntos seguidos).

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit/assets
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets/CS
$ (master)
$ cd ..
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets (master)
$ |
```

Nuevamente ingresaremos a una carpeta, ahora la carpeta **JS** a crear el archivo.

```

MINGW64:/d/Users/Catalina/Desktop/ProyectoGit/assets/JS
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets/CS
$ (master)
$ cd ..

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets (m
aster)
$ cd JS

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets/JS
(master)
$ touch logica.js

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets/JS (master)
$ |
  
```

Para continuar con los comandos básicos, eliminaremos la carpeta **IMG**, utilizando el comando **rm -r**. Para hacerlo debemos volver a la carpeta **Assets** y desde ahí eliminar el directorio. Corroboramos que se eliminó, listando los archivos.

```

MINGW64:/d/Users/Catalina/Desktop/ProyectoGit/assets
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets/CS
$ (master)
$ cd ..

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets (m
aster)
$ cd JS

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets/JS
(master)
$ touch logica.js

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets/JS (master)
$ cd ..

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets (master)
$ rm -r IMG

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets (master)
$ ls
CSS/ JS/

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/assets (master)
$
  
```

De esta forma hemos podido crear nuestro proyecto y en él cada uno de los archivos necesarios. Aprendimos a listar y eliminar directorios.

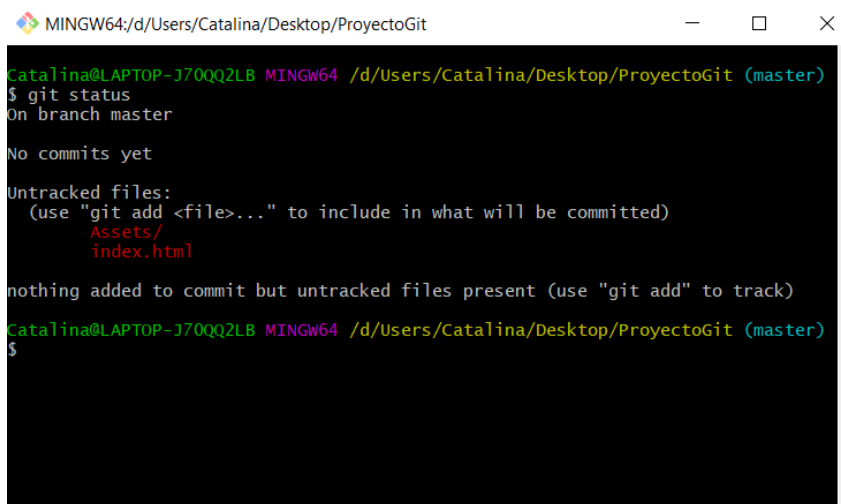
Para mayor orden, veamos una tabla con algunos de los comandos básicos de **Linux** que trabajamos:

comando	función
mkdir	Crear un nuevo directorio
rmdir	Eliminar directorio
rm -r (nombre archivo)	Eliminar directorio y su contenido
ls	Listar archivos
pwd	Encontrar la ruta del directorio en el cual nos encontramos
cd	Navegar por archivos y directorios
cat	Ver el contenido de un archivo
touch	Crear un nuevo archivo
mv	Mover un directorio
cp -r (nombre archivo)	Copiar un archivo

EXERCISE 2: COMANDOS BÁSICOS DE GIT

Ahora que hemos creado nuestro proyecto y sus archivos, vamos a comenzar a trabajar con los comandos básicos de **GIT**.

Previamente, utilizamos el comando **git init**, comando necesario para comenzar a “seguir el archivo” en el que estamos trabajando. Continuaremos con el comando **git status**, que nos permite ver el estado de los archivos y si están siendo seguidos o no.



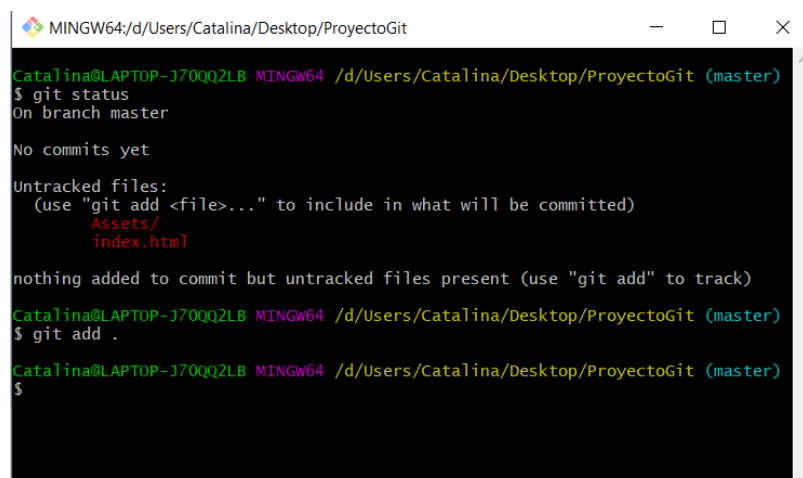
```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Assets/
        index.html

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

Acá podemos ver que aparecen en rojo el directorio **Assets** y el archivo principal **index.html**, indicando que ambos no están siendo seguidos y, al mismo tiempo, nos indica que debemos utilizar el comando **git add** sumado al nombre del archivo para poder comenzar a seguirlos.



```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Assets/
        index.html

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

También podemos seguir todos los archivos al mismo tiempo utilizando un punto, con el comando **git add .**

Para ver como se encuentran nuestros archivos ahora, usaremos nuevamente el comando **git status**.

```

MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
(use "git add <file>..." to include in what will be committed)
Assets/
index.html

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

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Assets/CSS/estilo.css
        new file:   Assets/JS/logica.js
        new file:   index.html

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$
  
```

Ahora podemos ver que los archivos aparecen en verde, esto significa que ya estamos siguiendo los archivos y podemos hacer un **commit**, es decir, guardar la primera versión de nuestro proyecto, el cual esta trabajando en la rama **master**.

Escribiremos el comando **git commit -m** y seguiremos con el texto que deseamos colocar como referencia para guardar esta versión.

```

MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Assets/CSS/estilo.css
        new file:   Assets/JS/logica.js
        new file:   index.html

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git commit -m "mi primer commit"
[master (root-commit) b62c704] mi primer commit
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Assets/CSS/estilo.css
create mode 100644 Assets/JS/logica.js
create mode 100644 index.html

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$
  
```

Para revisar nuestro **commit**, vamos a utilizar el comando **git log**, que nos mostrará la versión guardada.

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
(use "git rm --cached <file>..." to unstage)
new file:   Assets/CSS/estilo.css
new file:   Assets/JS/logica.js
new file:   index.html

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git commit -m "mi primer commit"
[master (root-commit) b62c704] mi primer commit
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Assets/CSS/estilo.css
create mode 100644 Assets/JS/logica.js
create mode 100644 index.html

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git log
commit b62c704ff14ca0105a37eed1a29911d7a9e76b0b (HEAD -> master)
Author: Catalina <m.catalina.gonzalez1@gmail.com>
Date:   Thu Dec 30 11:39:20 2021 -0300

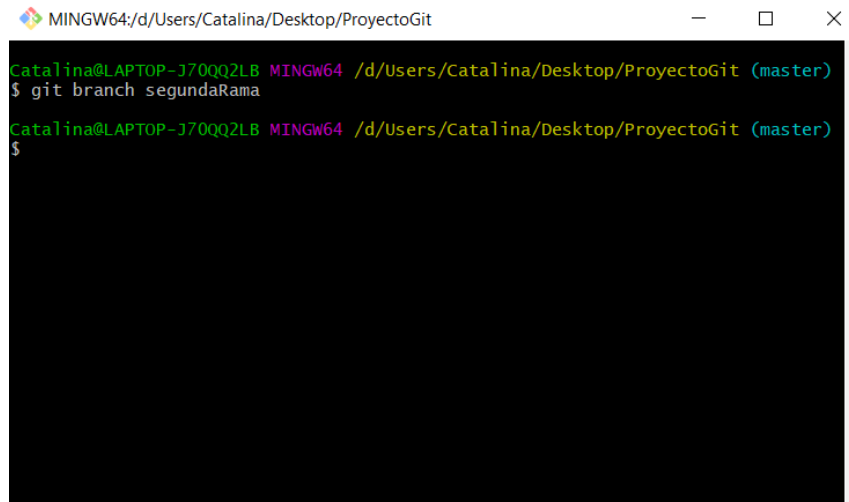
    mi primer commit

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$
```

CREANDO UNA RAMA

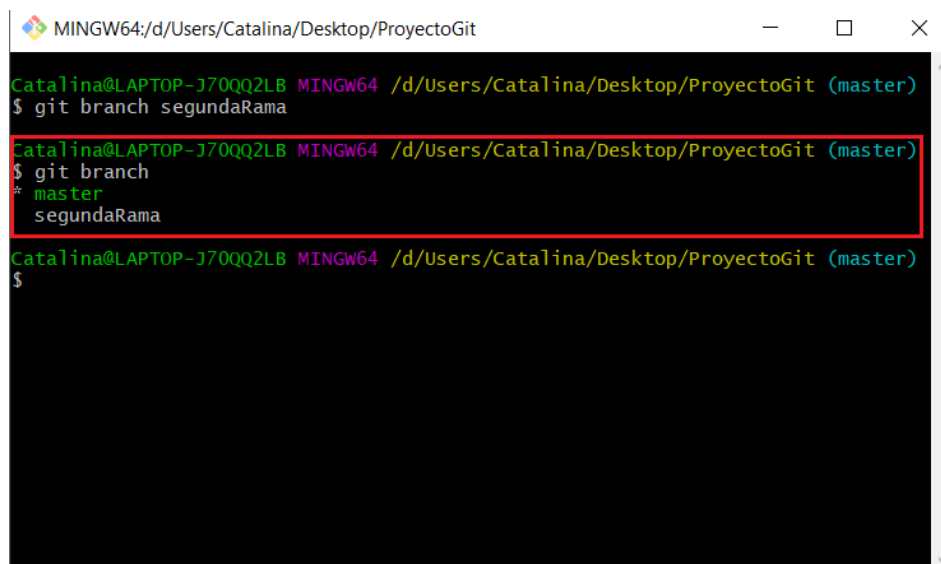
Cuando trabajamos con **GIT** para el versionamiento de nuestros proyectos, creamos ramas, es decir, distintas líneas en las cuales podremos tener distintos formatos de nuestro trabajo. Esto permite que el proyecto que se encuentra estable en la rama **master** (o rama principal) no sea afectado por los cambios que se puedan realizar en las otras ramas del proyecto, que podremos utilizar para probar nuevas funcionalidades y, una vez corroborado que están funcionando perfectamente, podemos fusionarlas con la rama **master**. Al mismo tiempo, los cambios que realicemos en la rama principal no afectaran al resto de ramas del proyecto.

Para crear una nueva rama, usaremos el comando **git branch**, seguido del nombre de la nueva rama.



```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git branch segundaRama
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$
```

Continuaremos colocando solo el comando **git branch** para ver las ramas que tenemos.



```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git branch segundaRama
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git branch
* master
  segundaRama
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$
```

Podemos observar que nos muestra que tenemos la rama “master” y “segundaRama”, al mismo tiempo, podemos ver que marca en verde la rama “master” indicándonos que es donde estamos posicionados.

Para movernos de una rama a otra, vamos a utilizar el comando **git checkout**, seguido del nombre de la rama a la que nos queremos mover.

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git branch segundaRama

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git branch
* master
  segundaRama

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git checkout segundaRama
Switched to branch 'segundaRama'

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (segundaRama)
$
```

Observemos como hemos pasado de la rama **master** a la **segundaRama**.

En la rama en la cual nos encontramos posicionados, volveremos a crear la carpeta **IMG** dentro de **Assets**.

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit/Assets

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git branch segundaRama

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git branch
* master
  segundaRama

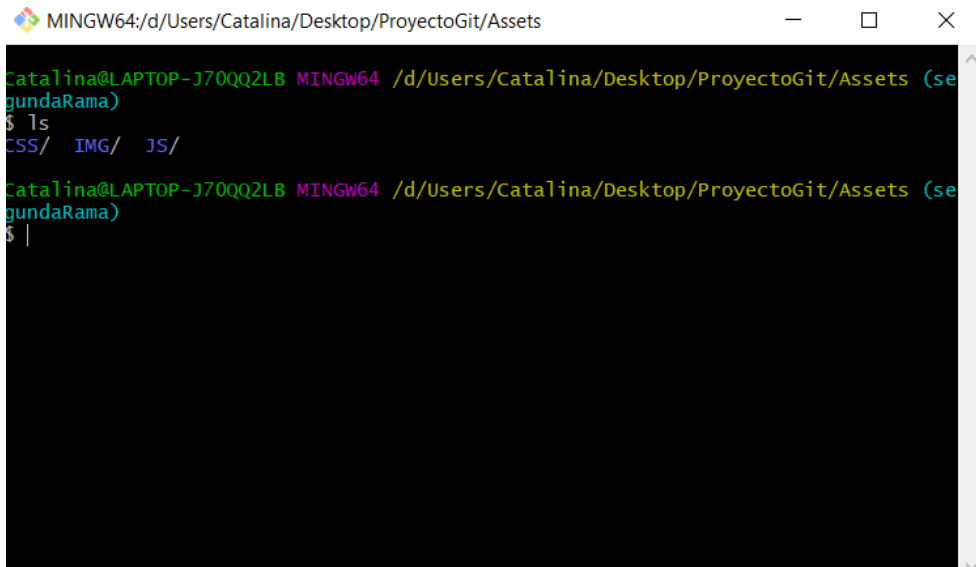
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git checkout segundaRama
Switched to branch 'segundaRama'

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (segundaRama)
$ cd Assets

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (segundaRama)
$ mkdir IMG

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (segundaRama)
$
```

Miremos que archivos se encuentran en **Assets**, usando el comando **ls**.

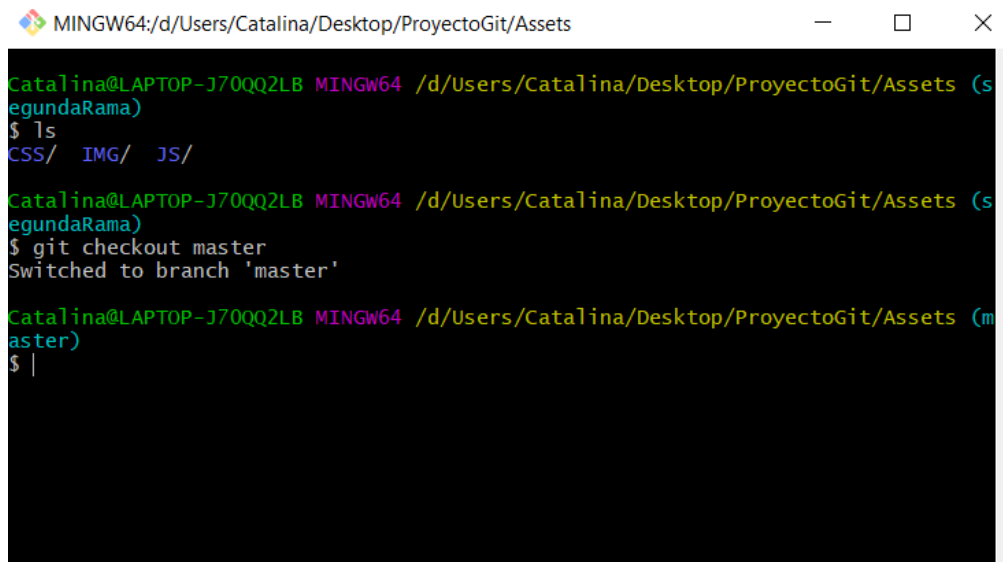


```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit/Assets

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (segundaRama)
$ ls
CSS/  IMG/  JS/

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (segundaRama)
$ |
```

Volvamos a la rama principal con **git checkout**.



```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit/Assets

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (segundaRama)
$ ls
CSS/  IMG/  JS/

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (segundaRama)
$ git checkout master
Switched to branch 'master'

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$ |
```

Retrocederemos a la raíz del proyecto con `cd ..`

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (segundaRama)
$ ls
CSS/  IMG/  JS/

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (segundaRama)
$ git checkout master
Switched to branch 'master'

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$ cd ..

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ |
```

Y posterior a eso vamos a colocar, en nuestro archivo principal, la estructura básica **HTML** (usando **Visual Studio Code**).

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-
7 scale=1.0">
8     <title>Document</title>
9 </head>
10 <body>
11
12 </body>
13 </html>
```


Vamos a revisar el estado de nuestros archivos con **git status**.

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
$ ls
CSS/  IMG/  JS/

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (segundaRama)
$ git checkout master
Switched to branch 'master'

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$ cd ..

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git status
On branch master
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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$
```

Podemos ver que nos muestra en rojo que se realizaron cambios en el archivo **index.html** y, al mismo tiempo, nos indica que acciones podemos realizar. Agregar los archivos con el comando **git add**, deshacer los cambios con el comando **git restore** o agregar y realizar el **commit** al mismo tiempo con el comando **git commit -a**. Para esta ocasión utilizaremos la tercera opción.

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
$ ls
CSS/  IMG/  JS/

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (segundaRama)
$ git checkout master
Switched to branch 'master'

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$ cd ..

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git status
On branch master
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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git commit -a
```

[illegible]

```
MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
$ git checkout master
Switched to branch 'master'

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit/Assets (master)
$ cd ..

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git status
On branch master
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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git commit -a
[master 96d5928] mi segundo commit
1 file changed, 12 insertions(+)

Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$
```

Ahora veremos nuestro **commit** usando **git log**.

```

MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
no changes added to commit (use "git add" and/or "git commit -a")
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git commit -a
[master 96d5928] mi segundo commit
1 file changed, 12 insertions(+)
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git log
commit 96d59287ced80ebe268210a3dea0c61349b0ec40 (HEAD -> master)
Author: Catalina <m.catalina.gonzalez1@gmail.com>
Date: Thu Dec 30 16:55:11 2021 -0300

    mi segundo commit

commit 70a923bde400865e877bd15445ce0dc42fd430ea (segundaRama)
Author: Catalina <m.catalina.gonzalez1@gmail.com>
Date: Thu Dec 30 16:34:39 2021 -0300

    mi primer commit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$
  
```

Si volvemos a la **segundaRama** y vamos a **Visual Studio Code**, veremos que los cambios realizados en el archivo **index.html** han desaparecido, ya que estos cambios se realizaron en la rama **master** y no en la **segundaRama**.

Para que haya más claridad con los comandos básicos de **GIT** vamos a evaluar la siguiente tabla:

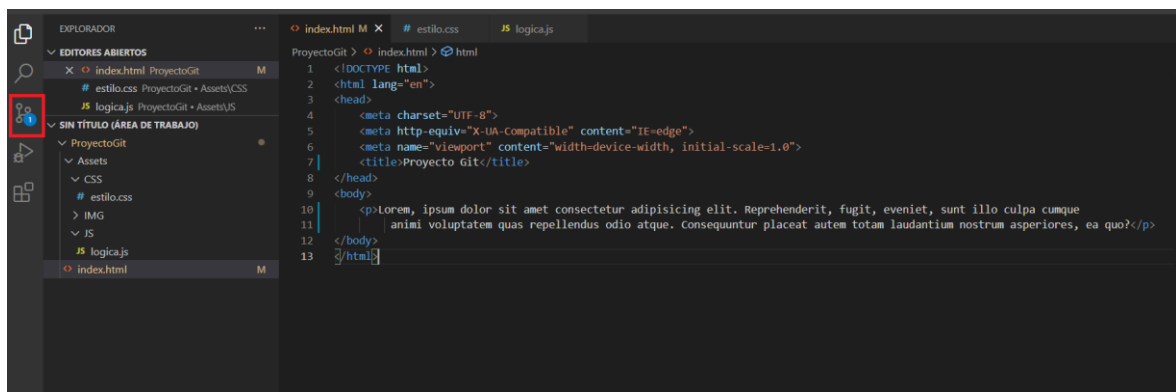
Comando	Función
git push	Cargar contenido a repositorio remoto
git init	Iniciar el repositorio
git status	Ver el estado de los archivos seguidos
git commit	Realizar el commit

git restore	Restaurar la versión anterior
git add	Agregar los cambios realizados
git log	Ver los commit realizados
git checkout	Ver las ramas
git branch	Crear una nueva rama
git diff	Comando para ver las diferencias realizadas en un archivo seguido

EXERCISE 3: GIT DESDE VISUAL STUDIO CODE

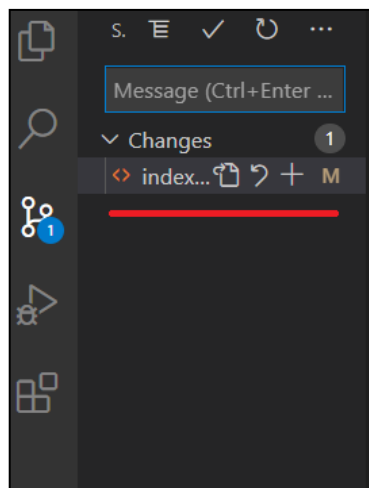
Hasta ahora hemos conocido los comandos básicos de **GIT** para trabajar desde la terminal, pero teniendo en cuenta que nuestro editor de texto es **Visual Studio Code** es importante saber que podemos realizar todos los mismos “movimientos” que hemos realizado desde la terminal directamente desde nuestro editor de texto.

Para comenzar este ejemplo, realizaremos cambios en nuestro proyecto en Visual Studio Code, cambiando el título y colocando una etiqueta <p> con Lorem Ipsum.

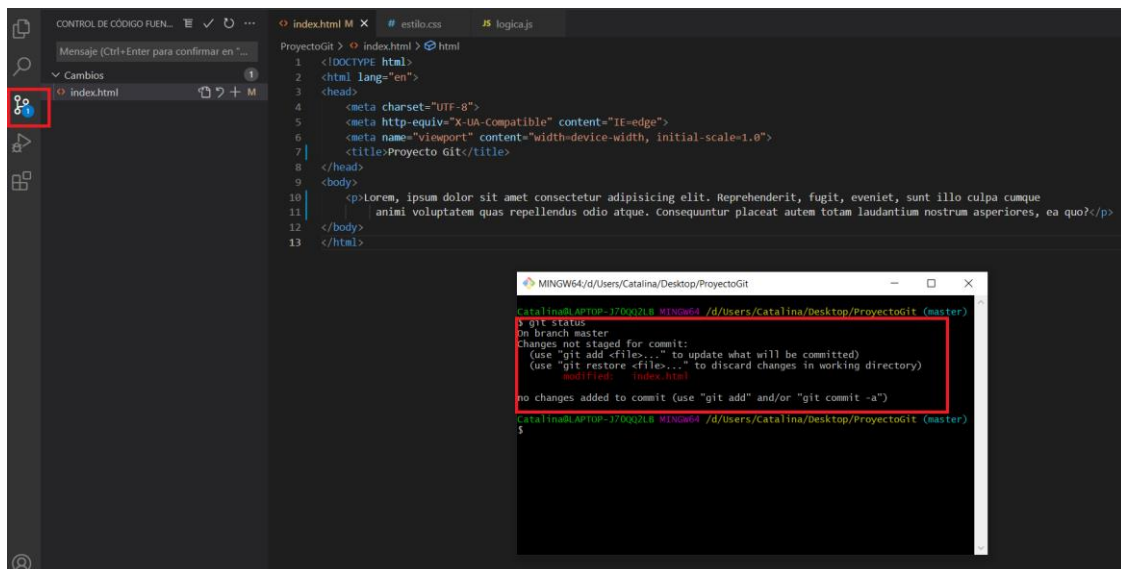


Al momento de realizar cualquier cambio, prestaremos atención al icono demarcado en rojo. Este nos mostrará inmediatamente un 1.

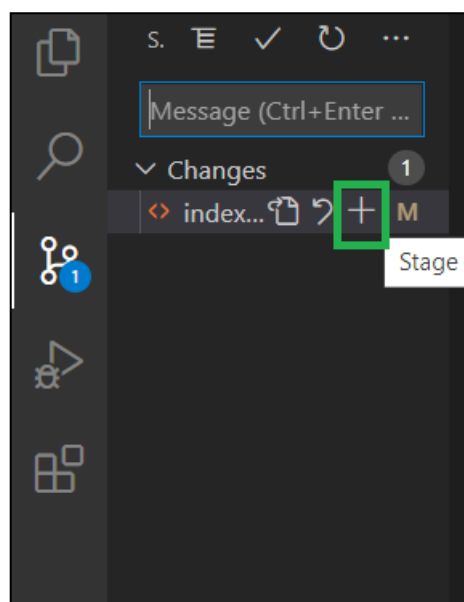
Hacemos clic en el icono y nos muestra un cambio.



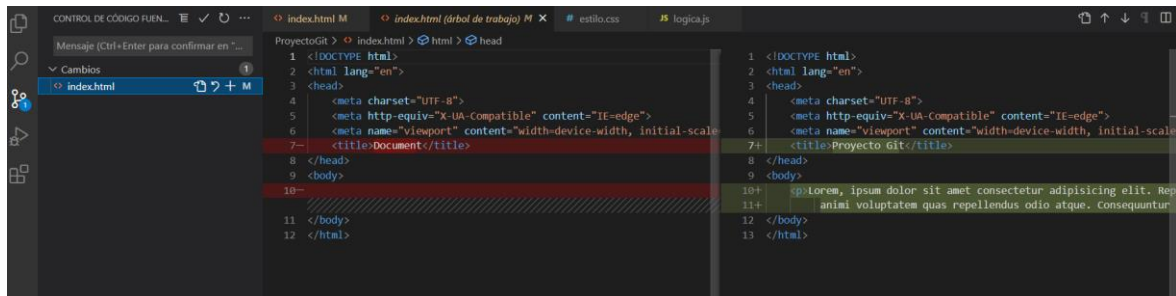
Lo podemos corroborar abriendo nuestra terminal de **Git Bash** y observando que nos indica que el archivo **index.html** ha sido modificado



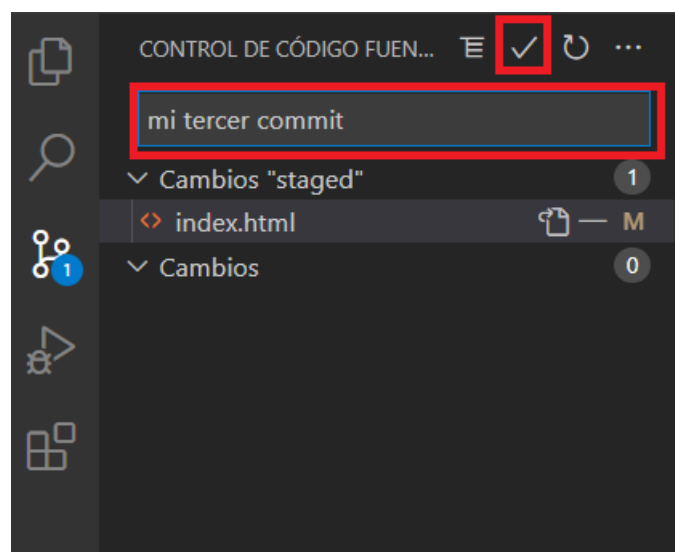
Para agregar el archivo, en la terminal debíamos escribir el comando **git add index.html**. Desde **Visual Studio Code** solamente debemos apretar el icono **más**.



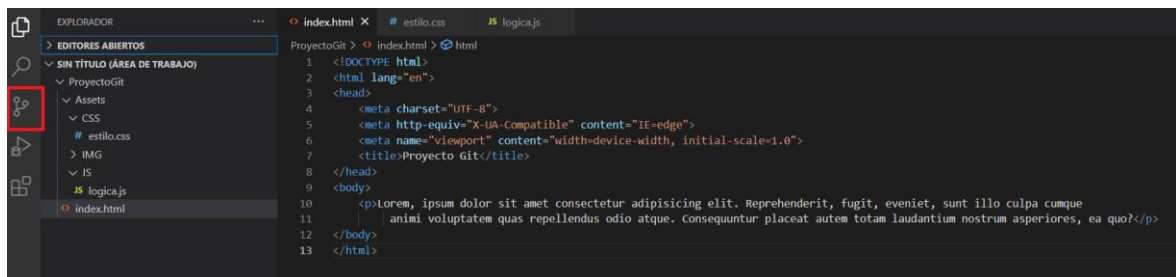
Una vez que hemos apretado el icono más, nuestro archivo pasa a estar en el listado de cambios preparados, listo para hacer el **commit**. Además, si apretamos el icono **M** podemos ver las diferencias entre este archivo y lo que había antes, mostrando en verde los cambios (lo que en la terminal deberíamos realizar con **git diff**) y apretando el icono de **menos**, podemos eliminar los cambios y volver un paso atrás.



Para hacer el **commit**, tenemos la posibilidad de escribir el mensaje descriptivo en la barra que dice **"Message"** y apretar el icono de **check** para confirmar el **commit** (sin tener que escribir ningún comando).



Podemos observar que posterior a esto nos deja de mostrar el cambio pendiente y podemos continuar trabajando.



Podemos corroborar este cambio en la terminal utilizando el comando **git log** y observar que ahora aparecen los tres **commit**.

```

MINGW64:/d/Users/Catalina/Desktop/ProyectoGit
no changes added to commit (use "git add" and/or "git commit -a")
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$ git log
commit daae5fd9ebd9ca024628954d5d74771f42633372 (HEAD -> master)
Author: Catalina <m.catalina.gonzalez1@gmail.com>
Date: Thu Dec 30 22:37:17 2021 -0300

    mi tercer commit

commit 96d59287ced80ebe268210a3dea0c61349b0ec40
Author: Catalina <m.catalina.gonzalez1@gmail.com>
Date: Thu Dec 30 16:55:11 2021 -0300

    mi segundo commit

commit 70a923bde400865e877bd15445ce0dc42fd430ea (segundaRama)
Author: Catalina <m.catalina.gonzalez1@gmail.com>
Date: Thu Dec 30 16:34:39 2021 -0300

    mi primer commit
Catalina@LAPTOP-J70QQ2LB MINGW64 /d/Users/Catalina/Desktop/ProyectoGit (master)
$
  
```

Después de esto podemos seguir practicando y conociendo mayores funcionalidades de **GIT**.