**Repositorio:** <https://github.com/PhilipeJoon/cursoFrontEnd>

1. **Necesidad de un repositorio de código fuente.**

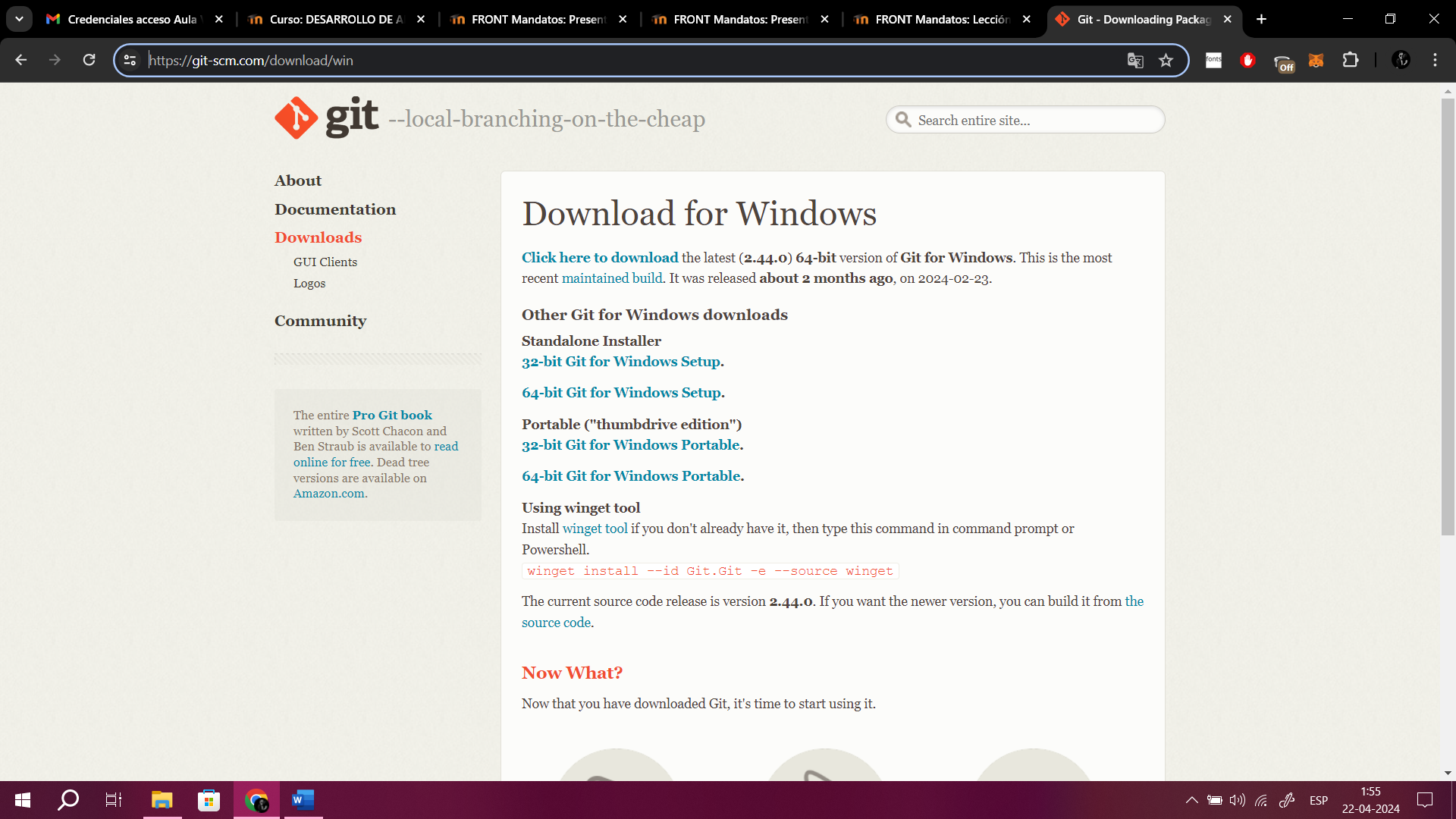
Un repositorio de código fuente es una herramienta del cual conviven los desarrolladores en sus labores, es importante su uso ya que de esta forma el código queda respaldado y se puede compartir a un grupo de trabajo sin mayores complicaciones para que puedan trabajar en torno al él.

1. **Instalación, configuración y comandos básicos.**

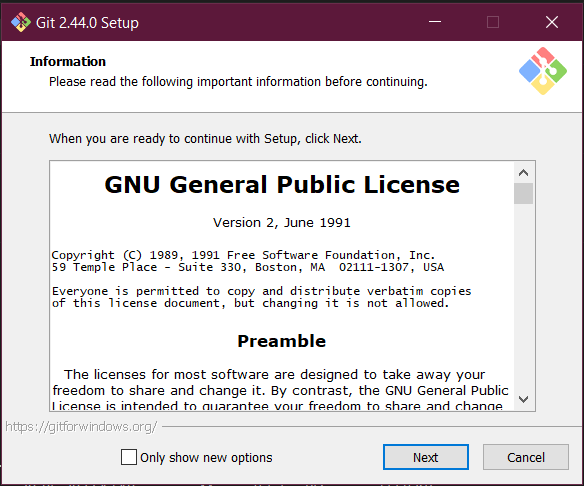
GIT es un software de control de versiones, por lo cual ayuda a administrar las diferentes versiones que se subirán del código a un repositorio de código fuente.

Debemos descargar el archivo de instalación de GIT desde la siguiente web <https://git-scm.com/download/win>

Se debe elegir la versión según si el sistema operativo es de 32 o 64 bit



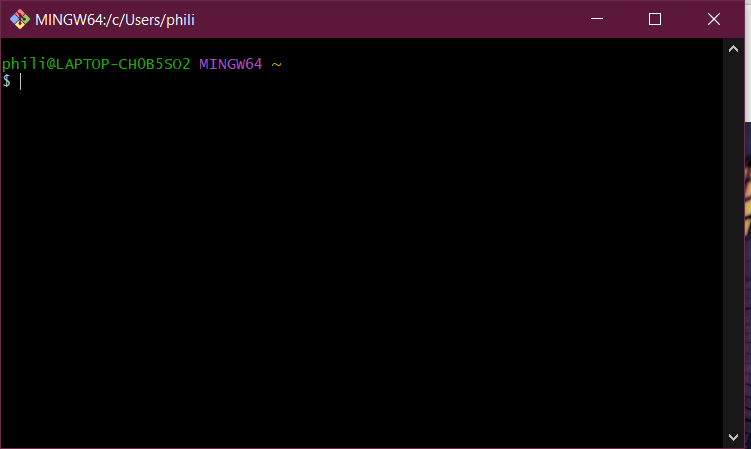
Una vez descargado, abrir el archivo y debe mostrar una ventana como esta



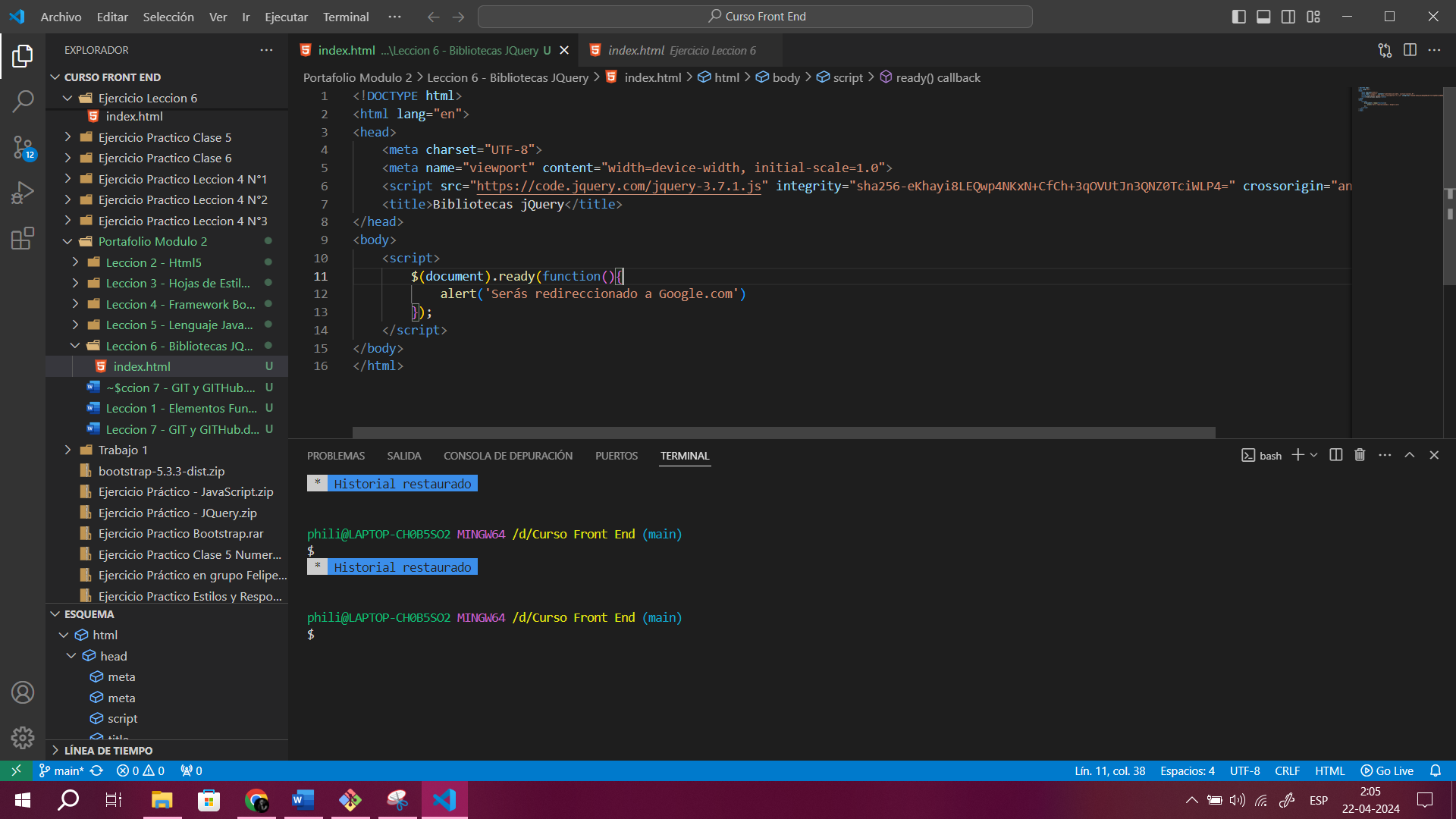
Presionar Next sin cambiar ninguna configuración y hasta completar la instalación. Una vez que este instalado, en el Escritorio del equipo vamos a inicio y buscamos Git Bash



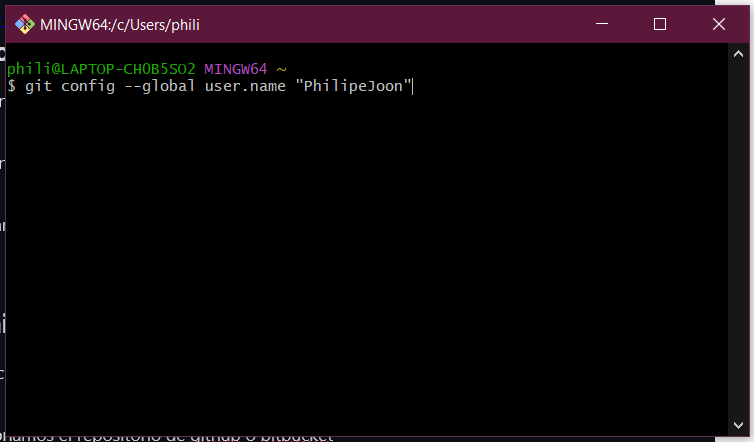
Abrimos Git Bash y nos debe mostrar una ventana como esta



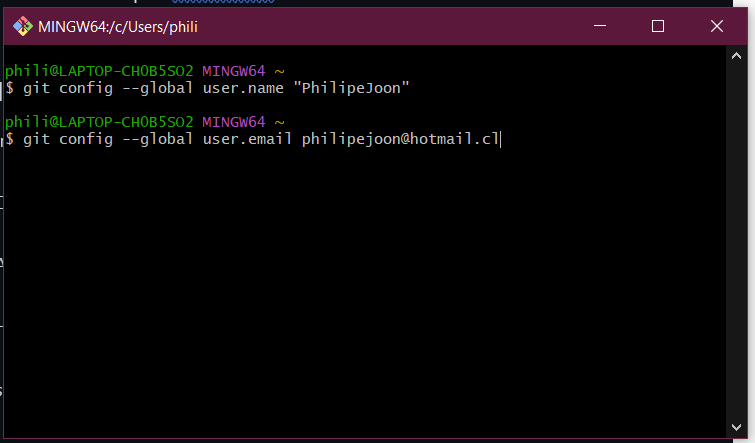
También tendremos habilitado este terminal en Visual Studio Code de la forma como se muestra en la siguiente imagen



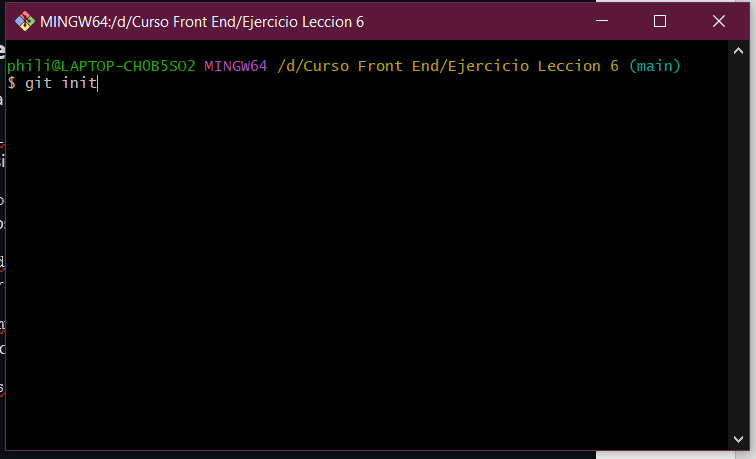
Una vez ya abierto GIT, se debe configurar la cuenta de GitHub en la cual tendremos todos nuestros repositorios. Primero hay que configurar el nombre de usuario de GitHub, el cual se realiza con el siguiente comando



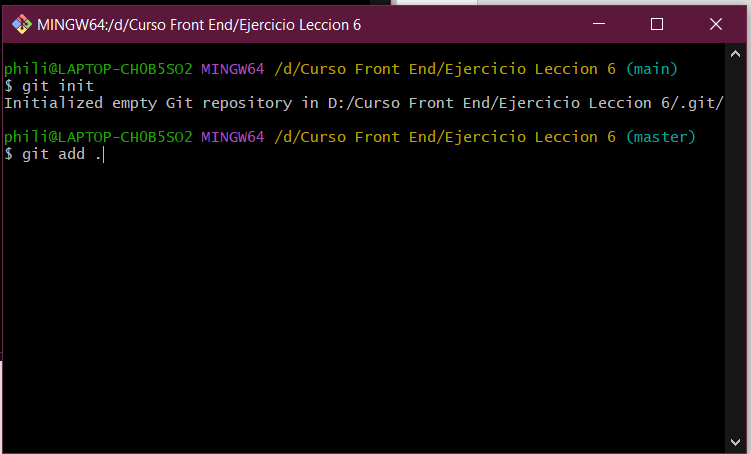
Presionar Enter y luego se debe ingresar el correo de la cuenta GitHub con el siguiente comando



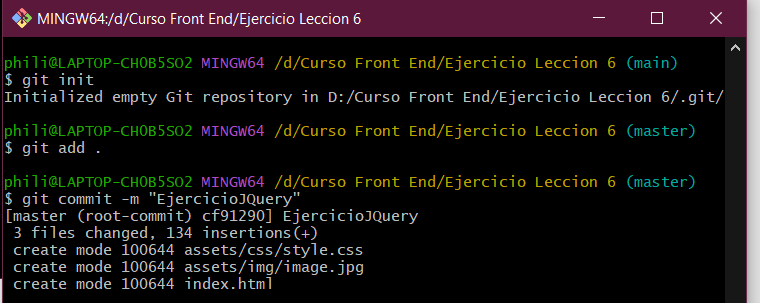
Presionar Enter y con ello ya tenemos configurada la cuenta, ahora hay que dirigirse a la carpeta donde esta el proyecto que se quiera subir al repositorio, y luego se ingresa el siguiente comando



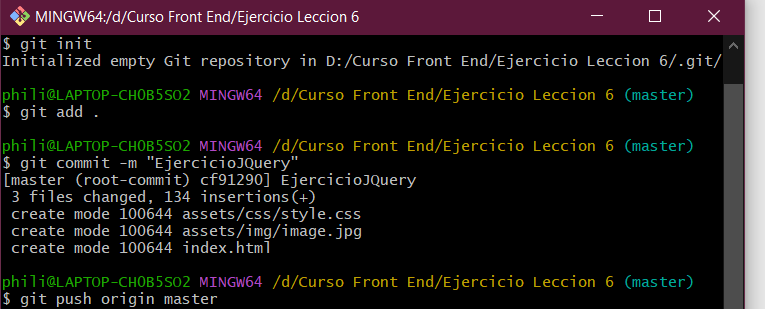
Presionar Enter y luego ingresamos el siguiente comando para agregar los archivos



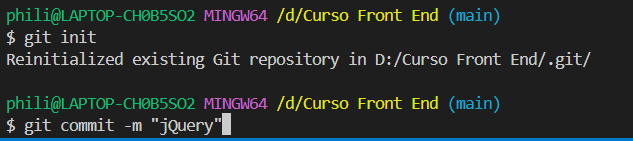
Presionar Enter, luego se debe crear el commit para subir los archivos al repositorio. Se crea con el siguiente comando. Cabe recalcar que entre las comillas debe ir el nombre que se le quiera poner al commit.

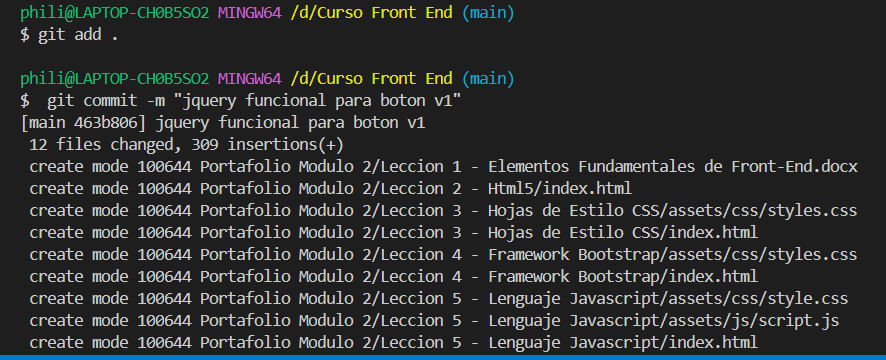


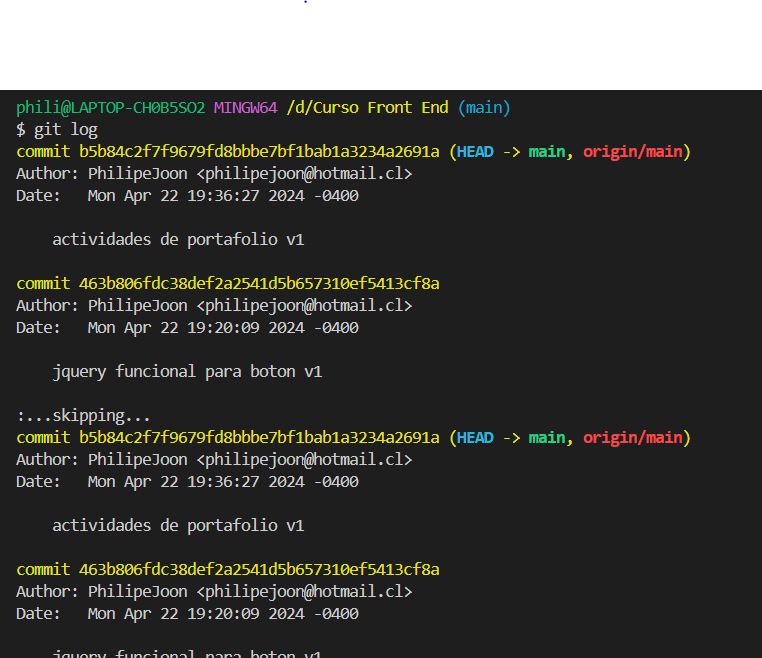
Y, por último, para subir todo al repositorio, se ingresa el siguiente comando



1. **Commits y restauración de archivos**

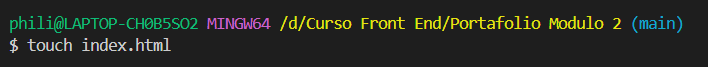




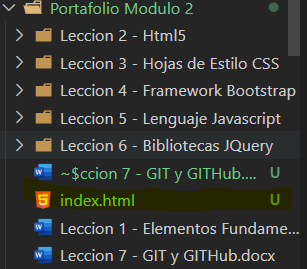


1. **Cambios de nombres**

Primero creamos un archivo desde la terminal de git bash con el comando touch



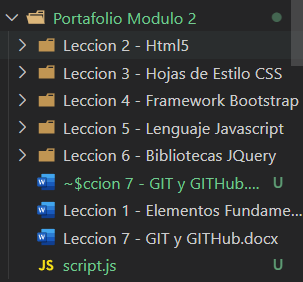
Verificamos que el archivo haya quedado en el directorio que elegimos



Siguiendo en la terminal de git bash, con el comando mv se cambia el nombre del archivo, como se muestra en la siguiente imagen



Por ultimo se confirma en el directorio que el cambio haya quedado realizado

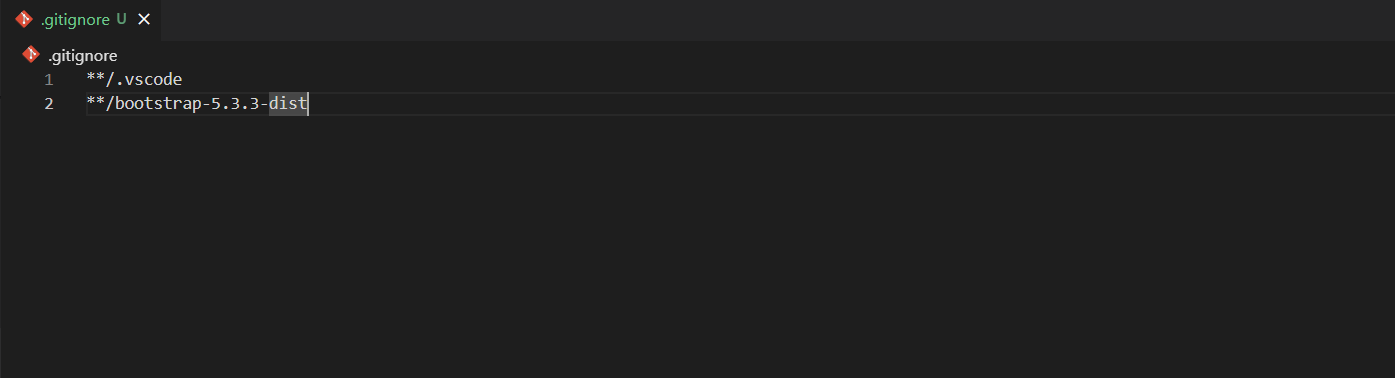


1. **Ignorando archivos**

Primero se debe crear el archivo .gitignore con el comando touch en la raíz del repositorio local

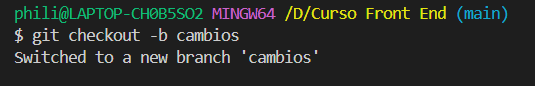


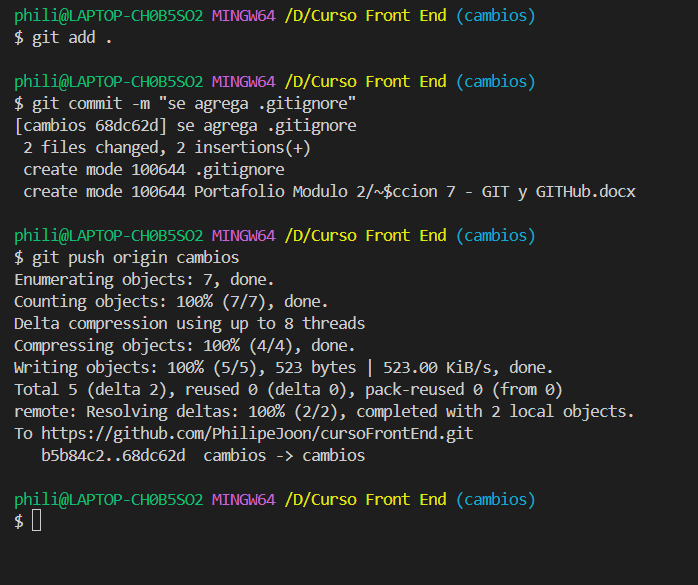
Se abre el archivo en el editor de Visual Studio Code, y dentro del archivo ingresamos los directorios que queremos ignorar

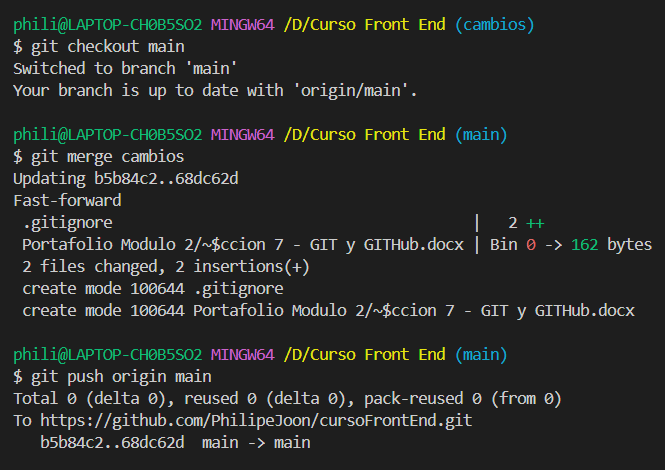


Lo que hace el archivo .gitignore es que al momento de subir un commit al repositorio en GitHub, no suba archivos y/o directorio al repositorio remoto.

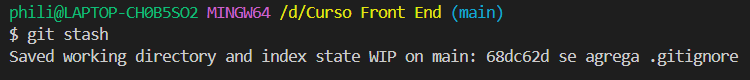
1. **Ramas, uniones, conflictos y tags**

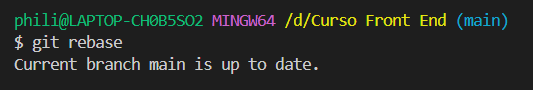




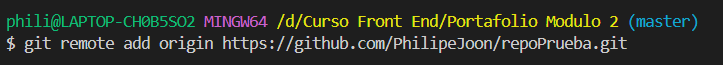


1. **Stash y Rebase**





1. **Repositorios remotos, Push y Pull**

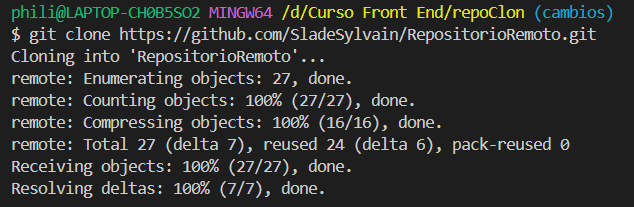


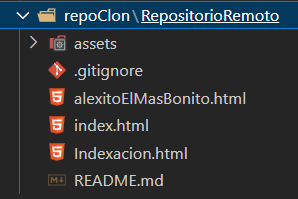
1. **Fetch vs Pull**

git fetch es un comando el cual descarga confirmaciones, archivos y referencias desde un repositorio remoto. Este no sobrescribe el repositorio local, solo es par ver y comparar los cambios hechos en el repositorio remoto con el local.

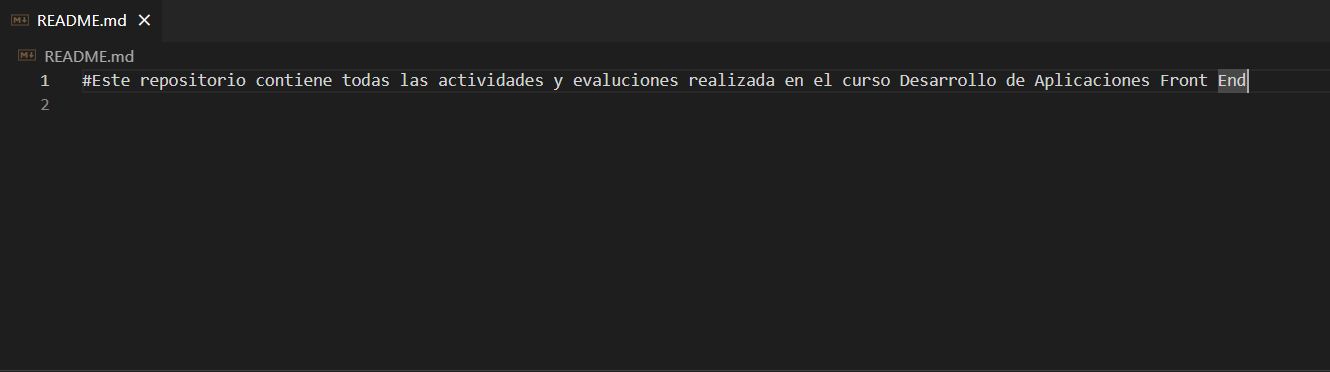
git pull es un comando que sirve para extraer y descargar contenido desde un repositorio remoto y actualizar al instante el repositorio local para reflejar ese contenido.

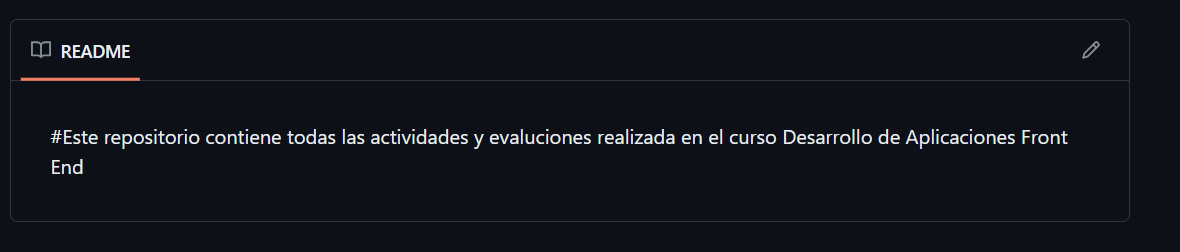
1. **Clonando un repositorio**



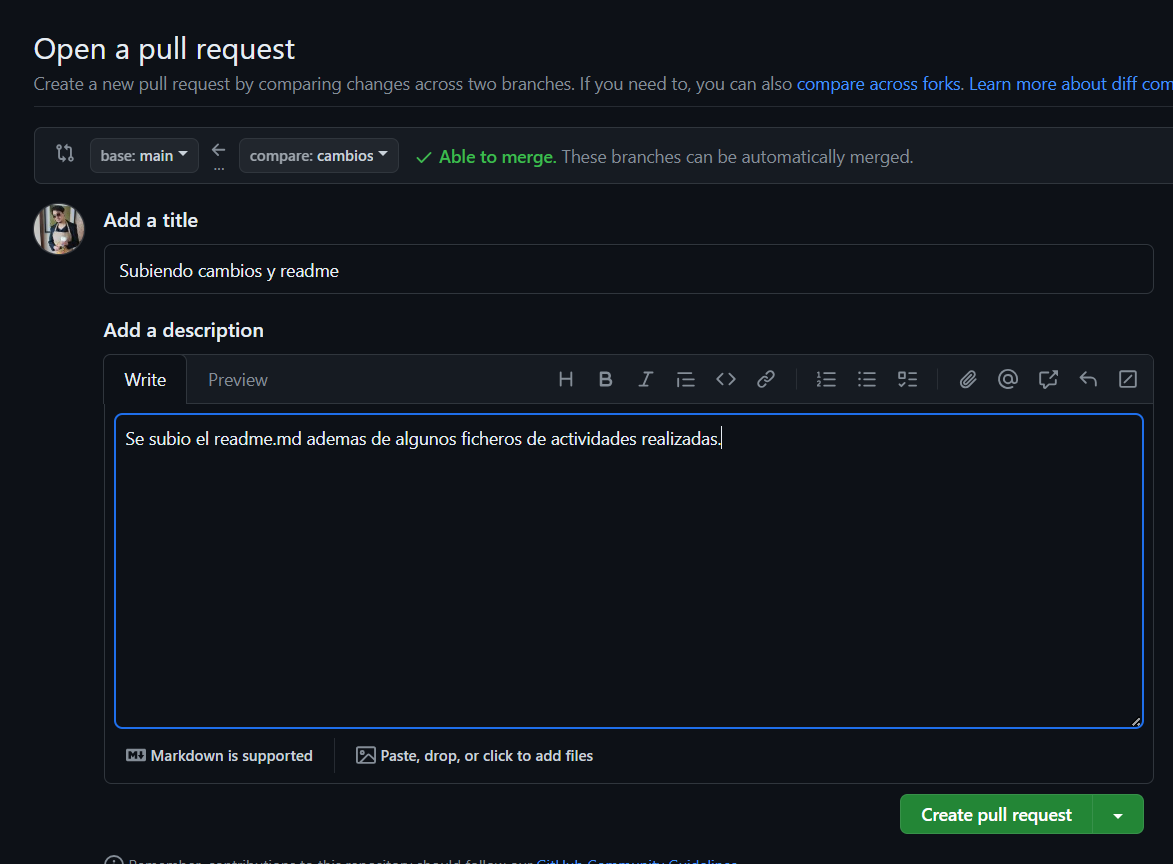


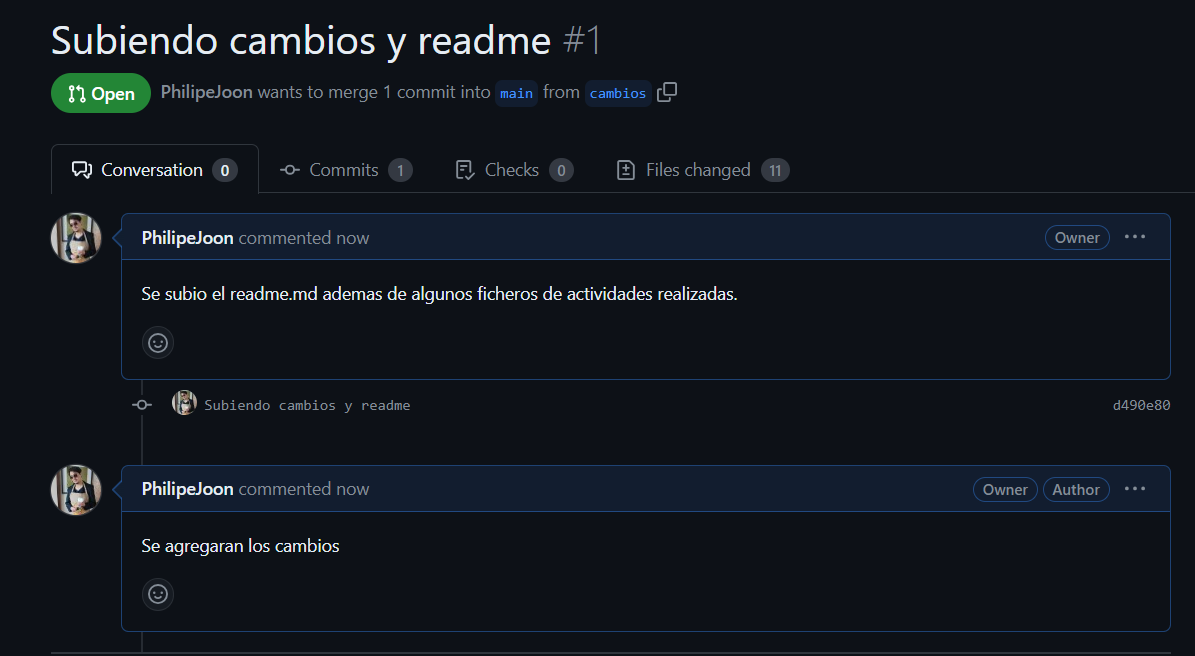
1. **Documentando un proyecto con Markdown**

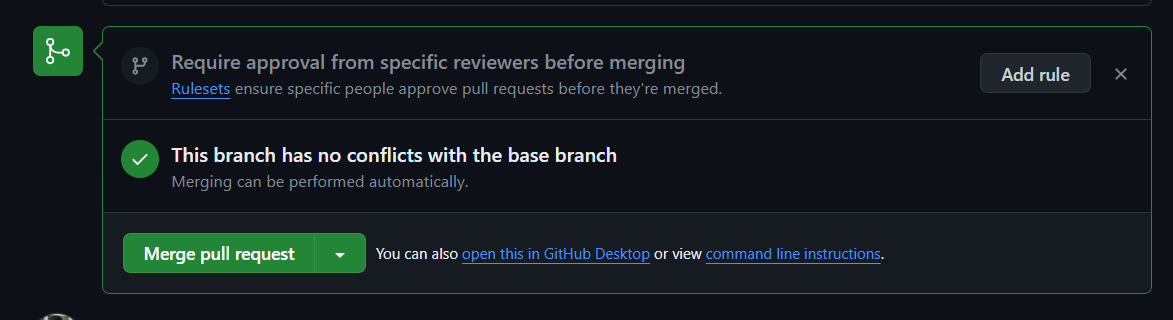


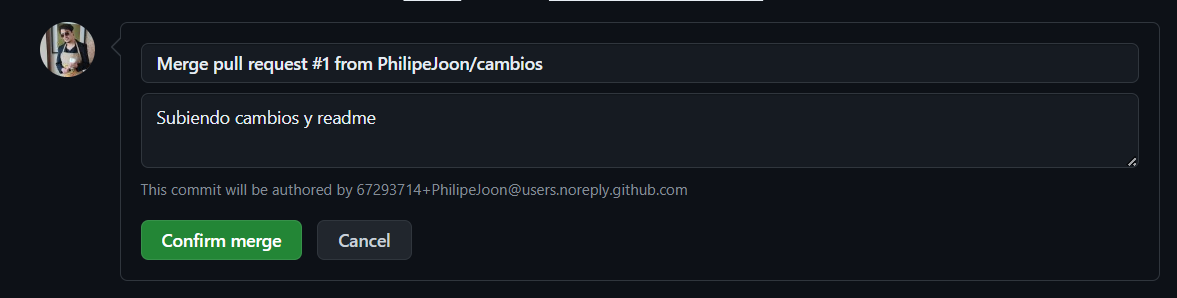


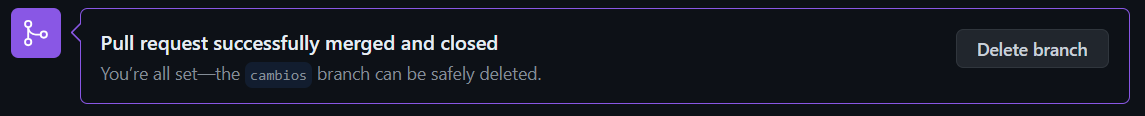
1. **Administrando Pull Request**











1. **Flujos de trabajo con GitHub**

Flujo de trabajo centraalizado: Es un flujo que solo se utiliza una rama, la cual es la main, y los cambios de todos los colaboradores se refleja en esa rama.

Ramas de funcionalidades: En este flujo se trabaja en otras ramas, dejando el código fuente en el main y todos los cambios se realiza en otra rama, la que luego de pasar todas las verificaciones se fusiona con la rama main.