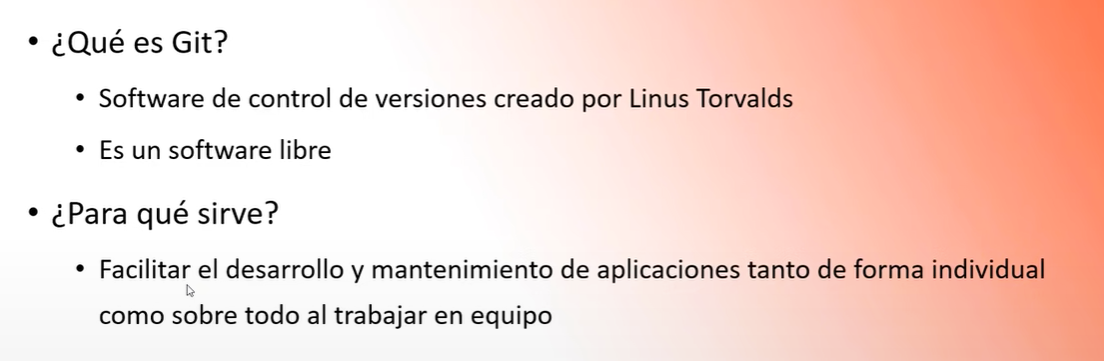
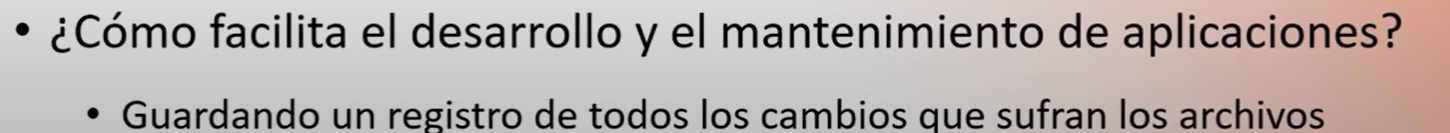
GIT & GITHUB

1. **TEMA 1: INTRODUCCIÓN**





GIT lo que hace es tomar una instantánea del archivo le añade un número de registro y una vez que hace eso nosotros podemos seguir programando. Git va tomando instantáneas según vamos programando y guardando en su registro interno



Cuándo tenemos algún problema con la aplicación que estamos programando por mil causas diferentes, es cuando podemos recurrir a GIT y a lo que ha ido almacenando en su registro interno

GIT déjame el archivo como lo tenía hace unos días. (Estado anterior ok). GIT no solo es capaz de trabajar con un archivo si no que es capaz de trabajar con muchos archivos (cuando trabajamos con un frameworks por ejemplo laravel para php sabemos que una aplicación está compuesta de muchos archivos)

GIT lo que hace es tomar una instantánea de los cientos de archivos que compone nuestra aplicación para así volver a un retorno de manera rápida y segura.



Otra funcionalidad es crear RAMAS (Trabajar en equipo sobre una aplicación o un archivo), consiste en dividir el archivo en tantos flujos de trabajo (Ramas)como programadores tenga el proyecto.

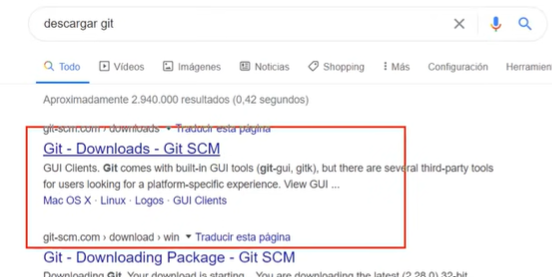
Cada programador se encarga de su parte de trabajo con sus instantáneas respectivas. Lo que GIT hace es fusionar los archivos de los programadores.

GIT también es capaz de detectar si dos o más programadores son capaces de trabajar en la misma parte del proyecto. Cuando GIT fusiona el archivo detecta que se ha hecho modificaciones en la misma parte del archivo por ejemplo tres programadores diferentes.

¿GIT nos pregunta que hacemos con esto???

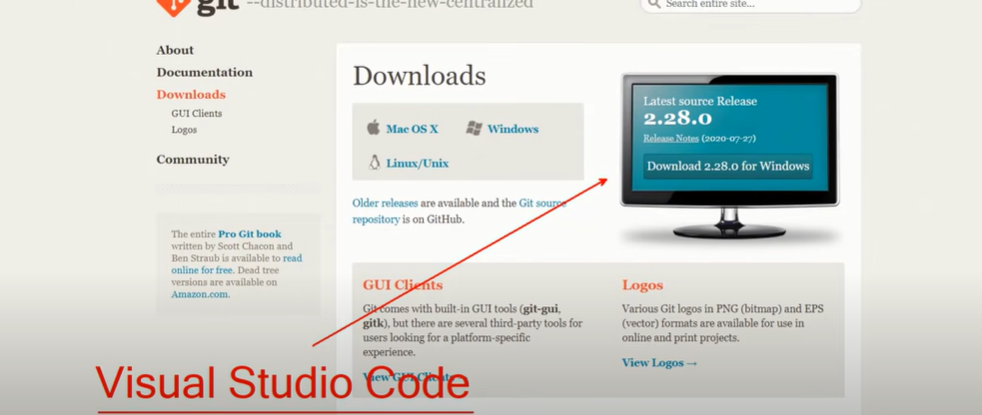
Nosotros le diremos lo que tenemos que hacer

Ahora vamos a descargar GIT

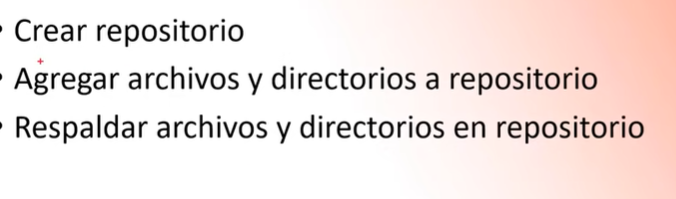


Cuando lo hayamos instalado lo más importante es la consola de GIT

GIT BaSH.

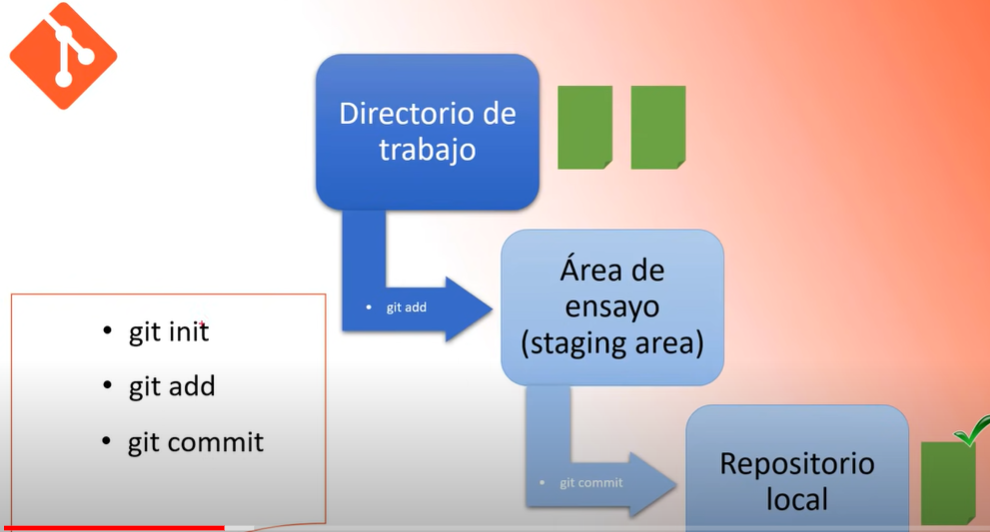
 Y luego tenemos un entorno de desarrollo que es capaz de trabajar en varios lenguajes que es capaz de trabajar de manera amigable con GIT.

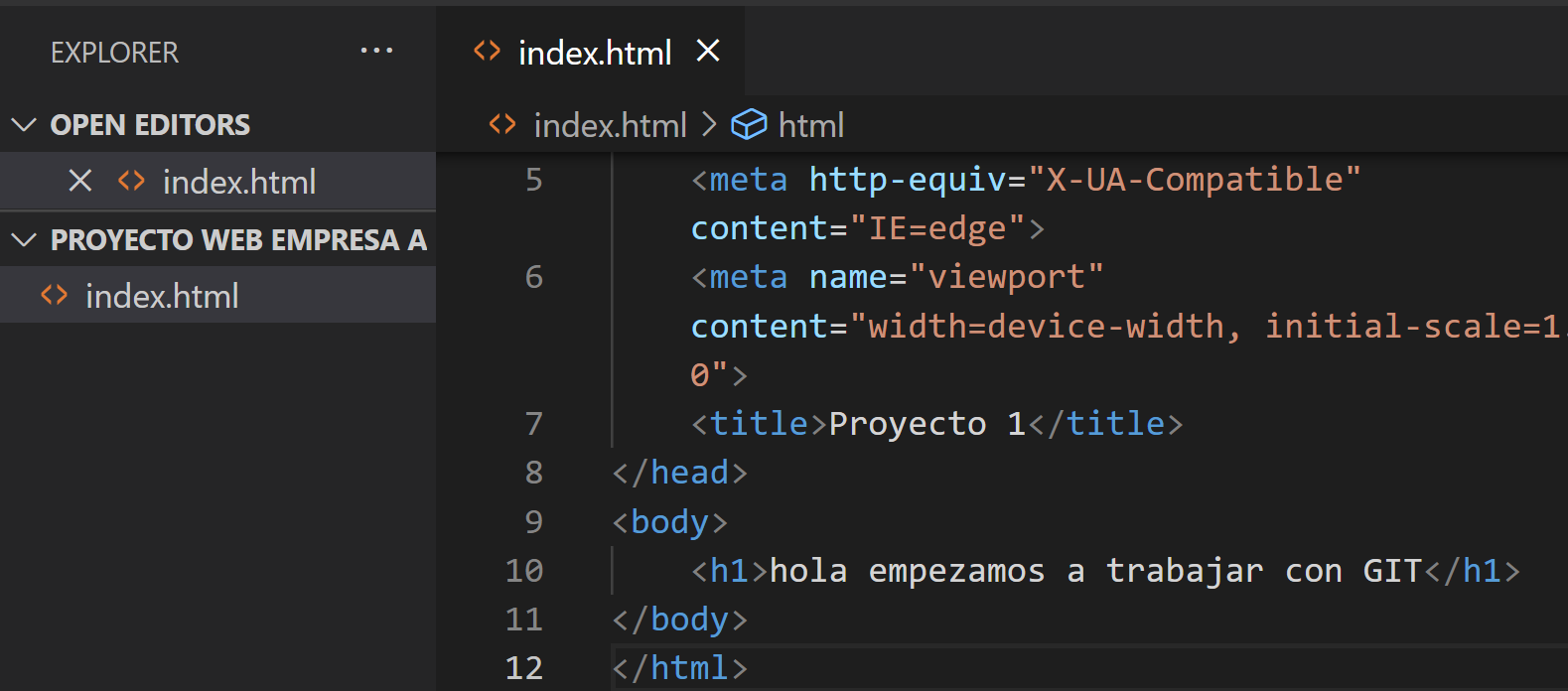
1. **TEMA 2: CREANDO REPOSITORIO**



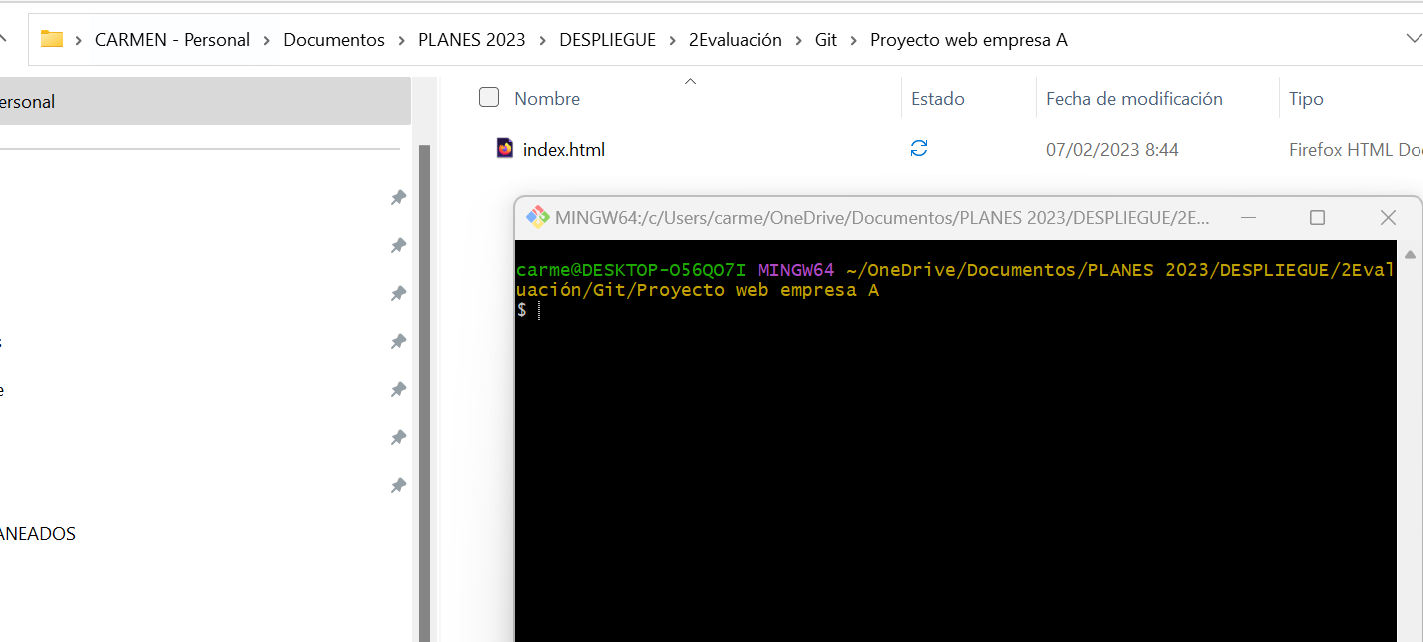
Para hacer estas tareas básicas con GIT, vamos a conocer los tres comandos básicos de consola:

* **git init:** cuando comenzamos el proyecto para hacer un seguimiento de el, lo que hacemos con esto es que git crea dos áreas (ÁREA DE ENSAYO (se almacenan los archivos de manera temporal, en qué estado se encuentran esos archivos) Y REPOSITORIO LOCAL (copias o instantáneas)
* git add: para que GIT haga un seguimiento a un archivo o archivos determinados y el archivo de seguimiento lo lleva al área de trabajo
* git commit: lleva el archivo del área temporal al repositorio local dónde GIT hace la instantánea.

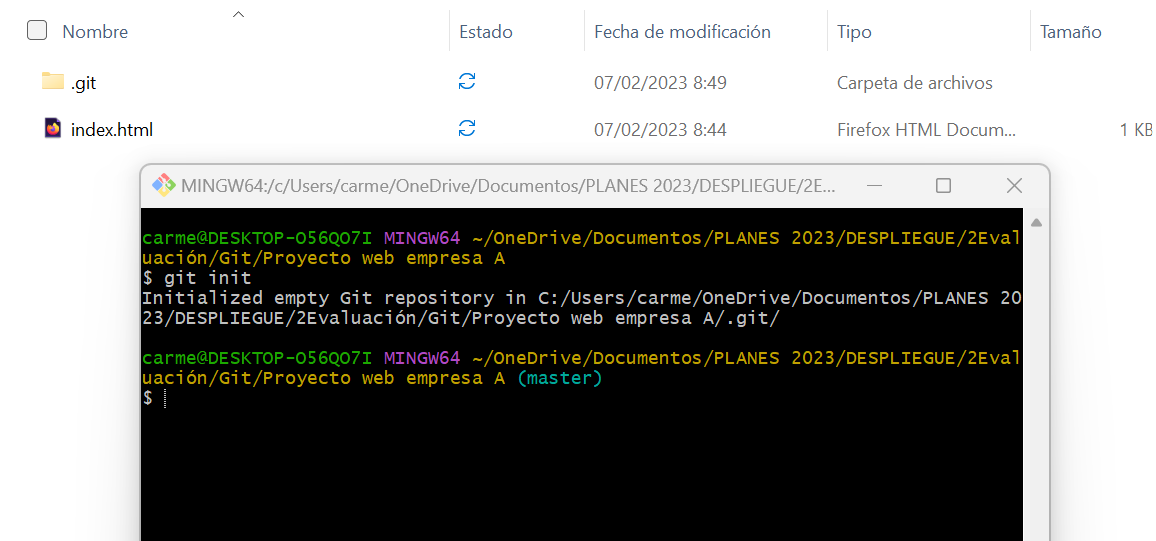




A partir de aquí quiero que GIT me haga un seguimiento y las copias de respaldo

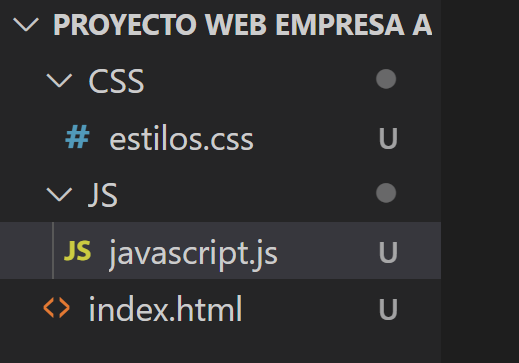


Ahora ponemos el comando git init (ya ha iniciado internamente el repositorio)

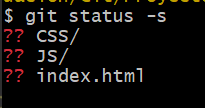


Decirle que muestre los archivos ocultos en Windows y veréis el repositorio

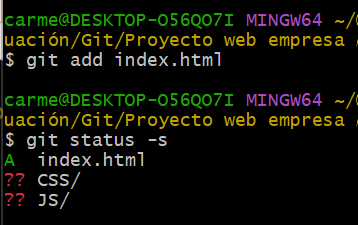
Volvemos a nuestro proyecto y creamos otras dos carpetas CSS Y JS con sus archivos correspondientes:



Para saber si GIT tiene constancia de todos los archivos y carpetas de nuestro proyecto:



Los interrogantes significan que ninguno está siendo sometido a seguimiento por GIT

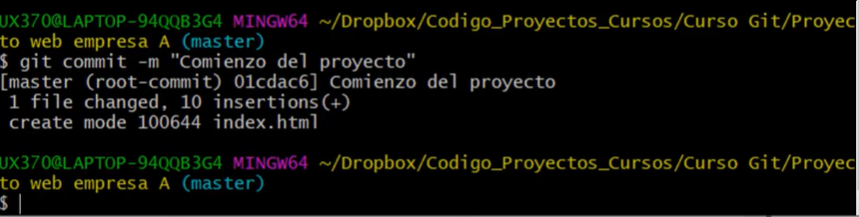
Vamos a decirle que haga un seguimiento a index.htmll Al archivo index ya lo está monitorizando(área de ensayo) y así nos lo indica la consola.

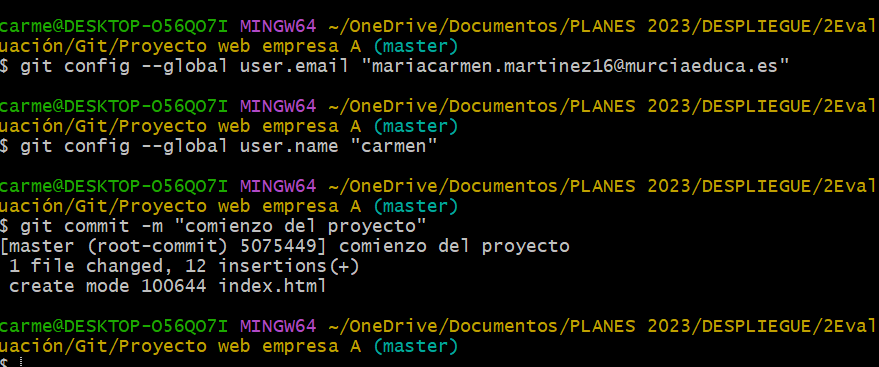
Para hacer una instantánea tenemos que hacer git commit y podemos agregar una descripción que nos va a ayudar bastante a saber los últimos cambios de nuestra copia de seguridad.

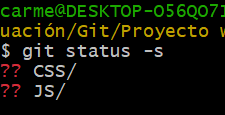
Si salen la primera vez unos mensajes de error, vamos a tener que utilizar

Usernamegi

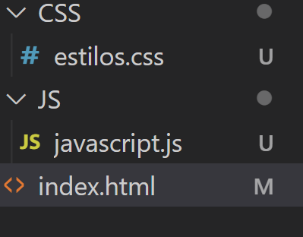
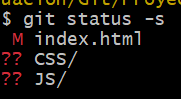
email

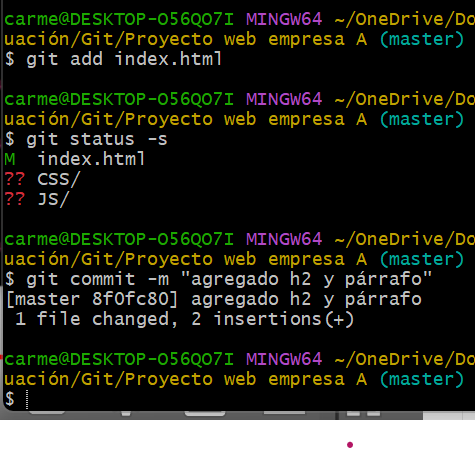






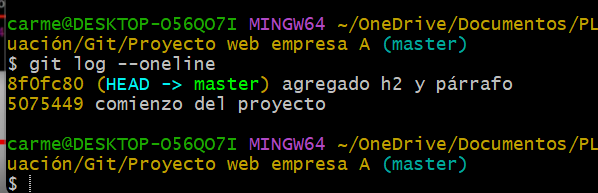
Si a nuestro archivo index le hacemos alguna modificación

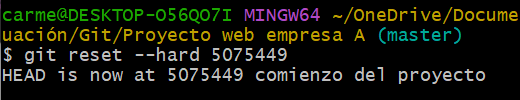


Hay que darse cuenta que pasa M del estado rojo al verde

¿??Cuántas instantáneas tengo del archivo index ¿?? Tantas como commit haya hecho. Para verlo:

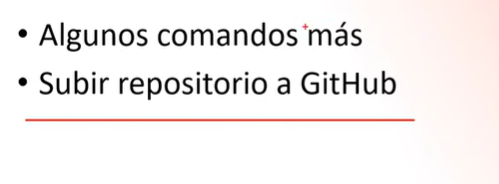


Ahora si quiero una restauración a lo que tenía al principio la primera instantánea



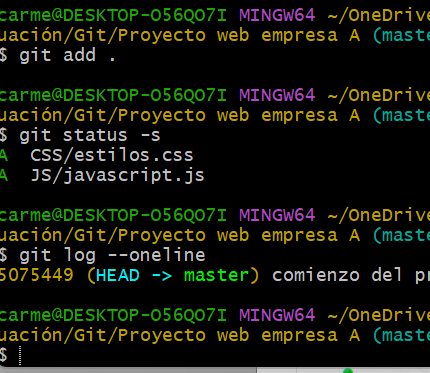


1. **TEMA 3: SUBIENDO GITHUB**

****

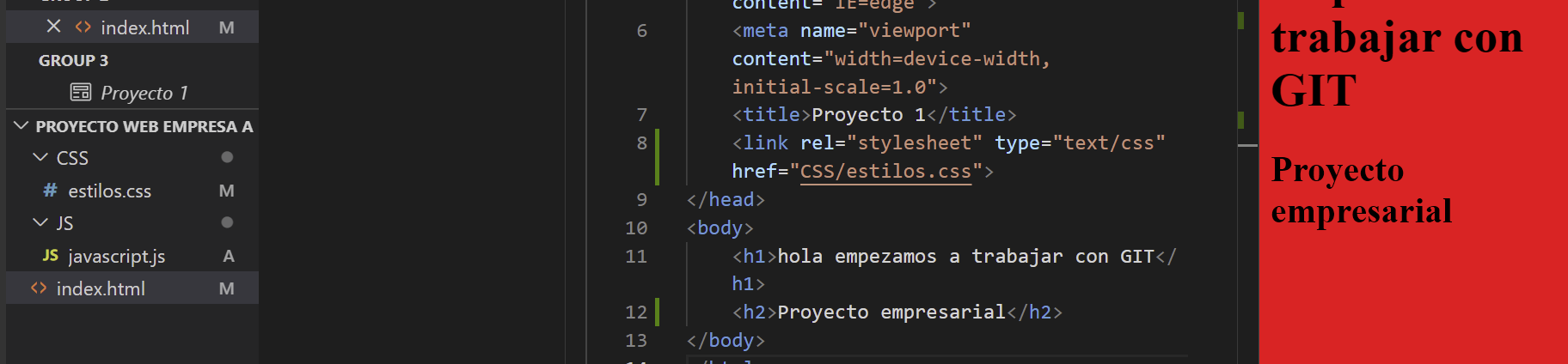
Es importante que no esté todo nuestro proyecto en Git (local) y así subirlo a la nube (GITHUB)

Vamos a ver cómo hacer un seguimiento a todos los archivos de nuestro proyecto, agregamos al área y vemos que le está haciendo seguimiento a estos dos archivos A

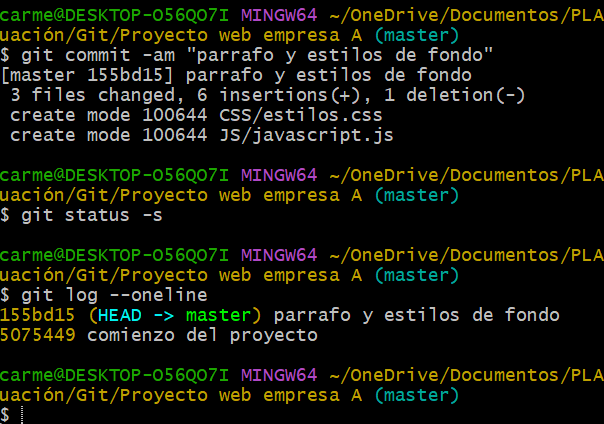


Para ver los commit que tenemos guardados en este proyecto sería con git log –oneline. Vamos a agregar un segundo proyecto.

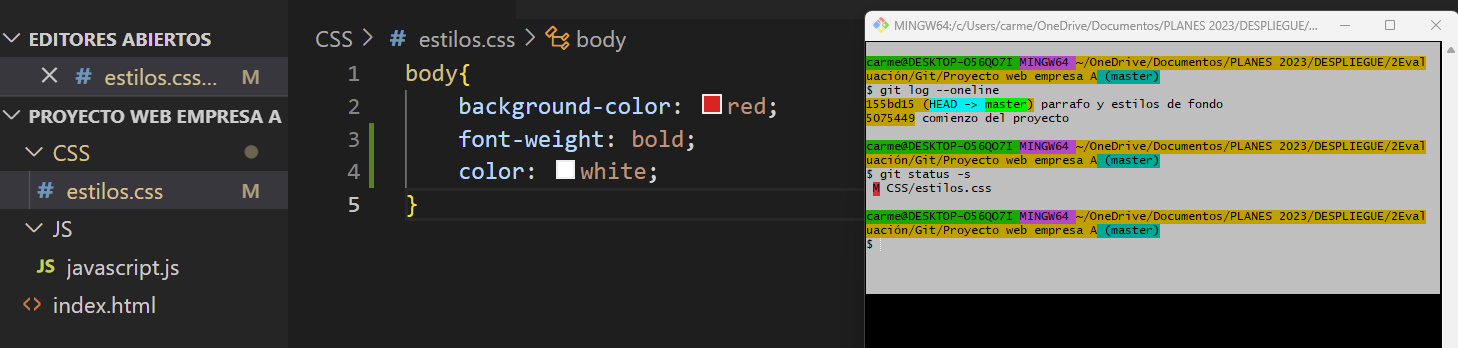
Vamos a hacer cambios en nuestro proyecto (hoja de estilos) por ejemplo un color de fondo

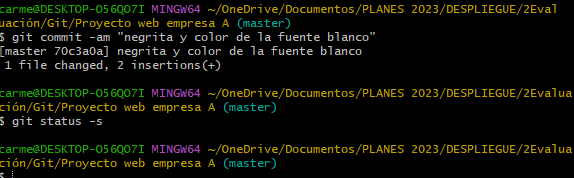


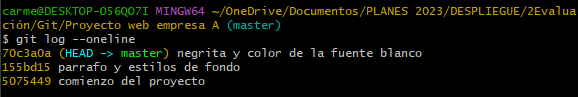
Como es muy frecuente cuando estamos haciendo cambios hacer un add y commit a la vez podemos hacer las dos operaciones



Si hacemos un status no aparece nada pq ya tenemos todo añadido al repositorio. Vamos a hacer un commit más para darle más intensidad a nuestro proyecto para eso le añado



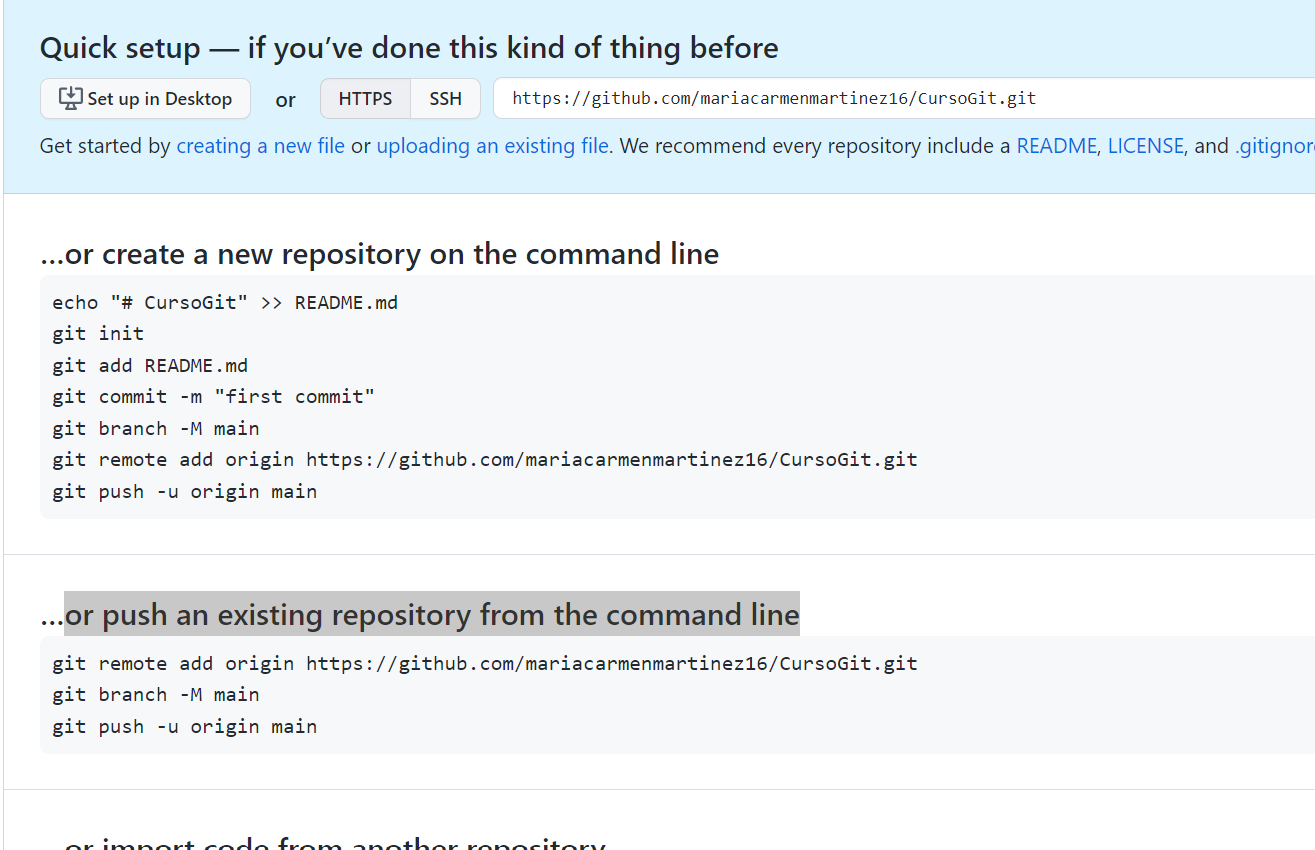




Ya tenemos los tres commit

AHORA VAMOS A VER COMO SUBIR NUESTRO PROYECTO EN LA NUBE PARA TENER UNA COPIA DE RESPALDO.

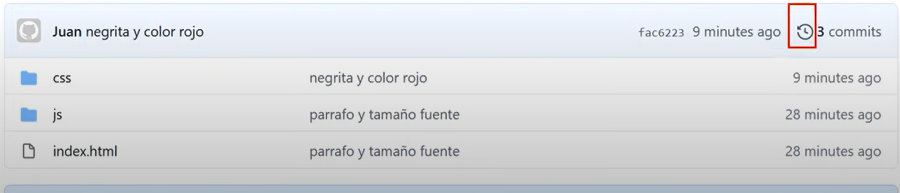
GitHub.com (darse de alta) Nombre de usuario y correo electrónico



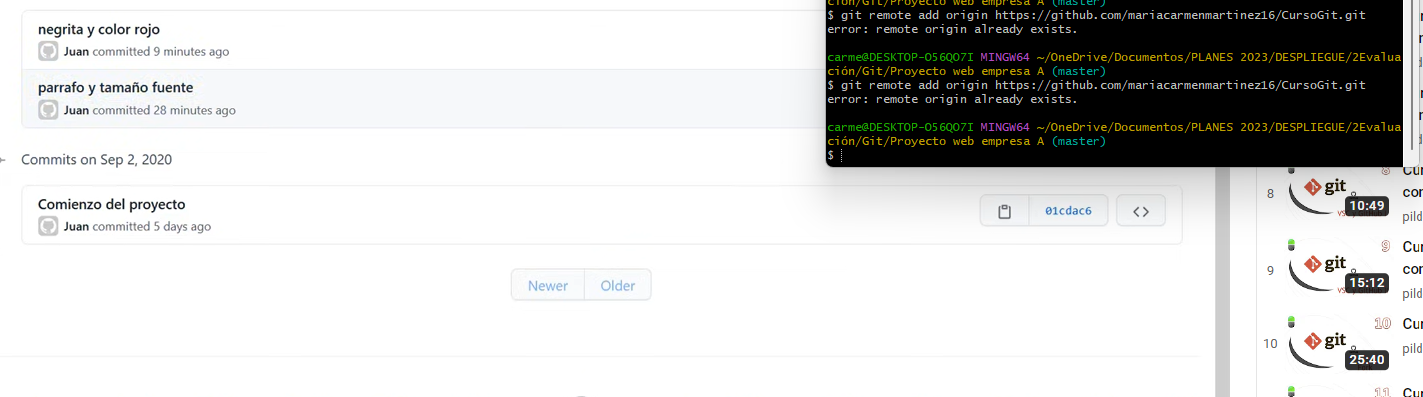
Podemos elegir public o private y en nuestro caso el tipo de repositorio es el que está seleccionado repositorio existente y lo vamos a subir desde nuestra línea de comandos.

Nos vamos a fijar en esos dos comandos que tengo seleccionados con flechas

Si seleccionamos la primera línea y la copiamos en nuestra consola subirá todo lo que tenemos en local. Actualizamos y debe salir nuestro primer repositorio



Si pulso donde aparece los 3 commits que he realizado o registrado en cada uno de mis proyectos y desde GitHub puedo ver el estado de cada uno de mis proyectos.

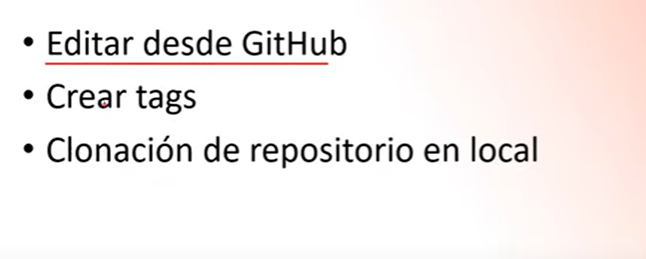




No dirá como estaba el proyecto en ese instante

¿¿¿¿ Cuántos repositorios podemos crear????? De momento no hay límite

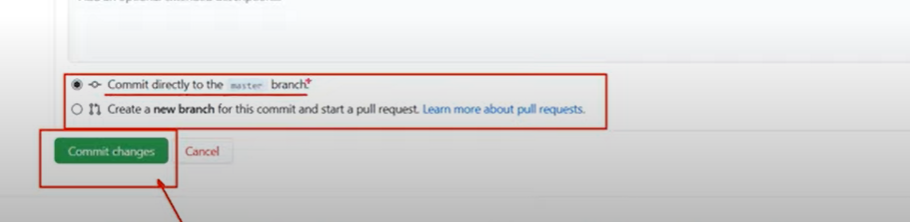
1. **TEMA 4: CLONACIÓN Y TAGS**

****

Si deseamos programar desde GitHub lo podemos hacer desde esta ventana



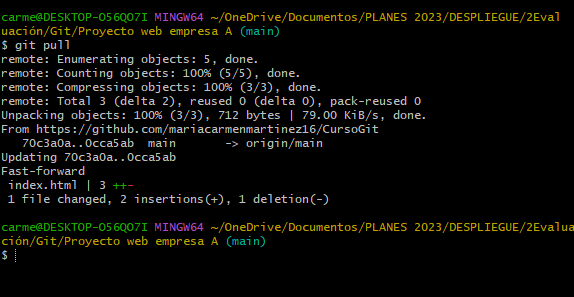
Puede ser útil para hacer pequeñas modificaciones y haces el proceso a la inversa. Un poco más abajo aceptamos los cambios



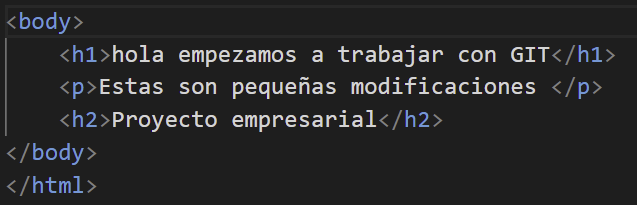
Con esta opción hemos guardado los cambios en remoto

Estos cambios no están en local. ¿Qué tenemos que hacer ¿???????

Utilizamos la instrucción pull

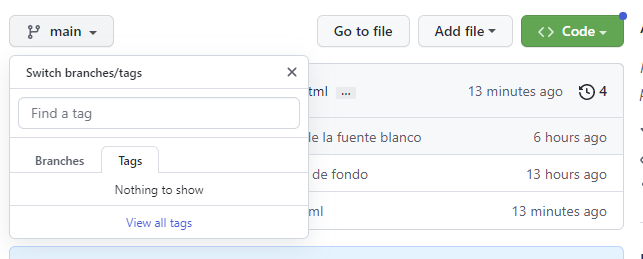
detecta los cambios automáticamente.

Se añade automáticamente en local

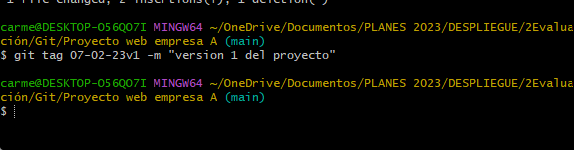


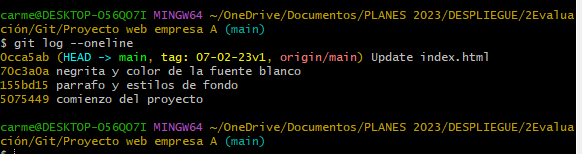
Si queremos guardar nuestro proyecto como una primera versión 1.0 que nuestro proyecto ya está acabado. Aunque en un futuro tenga mejoras.

¿Cómo le decimos que todos los miembros de este grupo de trabajo y cualquier otra persona como es un repositorio es público se pueda descargar esa primera versión. Esto se hace gracias a las TAGS(ETIQUETAS) NOS PERMITE ETIQUETAR VERSIONES DE NUESTRO PROYECTO.

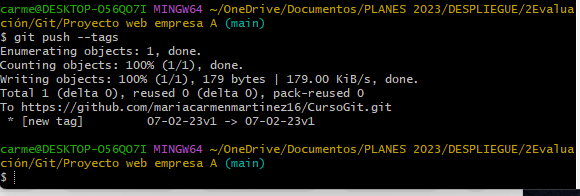


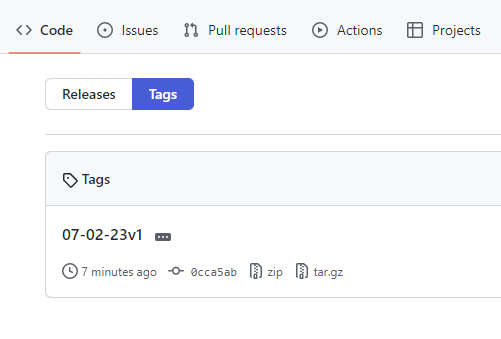
Me dice que Nothing to show en los Tags . Nos vamos a consola para crear mi primer tags





Git ha tomado nota y así lo vemos con el comando git log –oneline. ¿Qué hay q hacer para que este tags que está en local aparezca en remoto??????

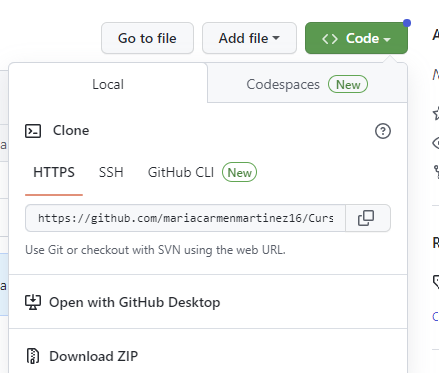




En GITHUB ya tengo el primer Tags (que me da la opción de descargarme el proyecto en zip o en tar si estoy en Linux).

CLONACIÓN DE REPOSITORIO EN LOCAL

Simulamos un desastre y eliminamos la carpeta del proyecto



Copiamos la URL del proyecto y nos vamos a consola

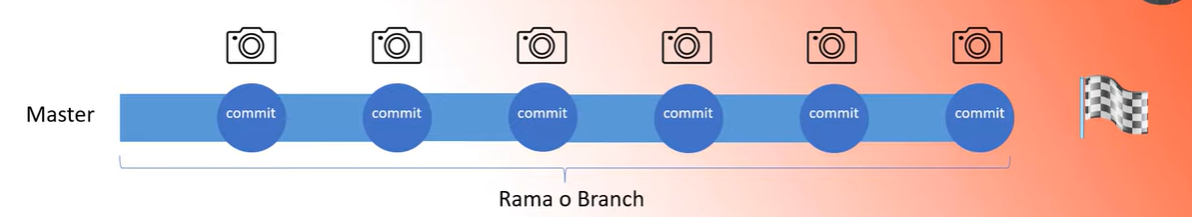


1. TEMA 5: RAMAS O BRANCHES I

RAMAS O BRANCH

Ramas o líneas de tiempo durante el desarrollo del proyecto desde el inicio al final le llama Rama Master o main.

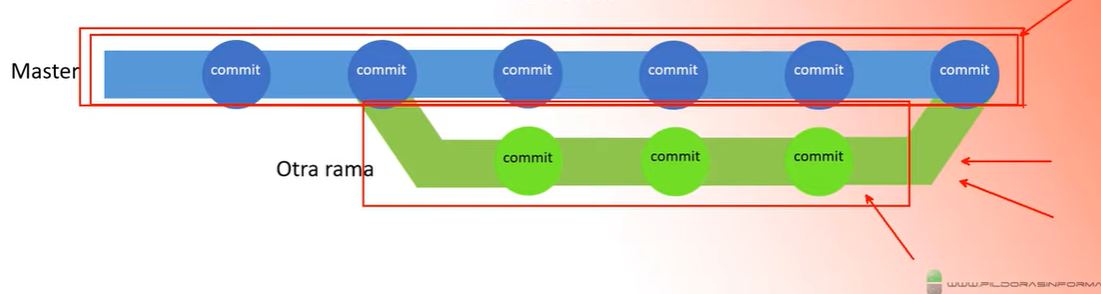
Con git podemos crear varias ramas paralelas o varias líneas de tiempo de los proyectos complejos



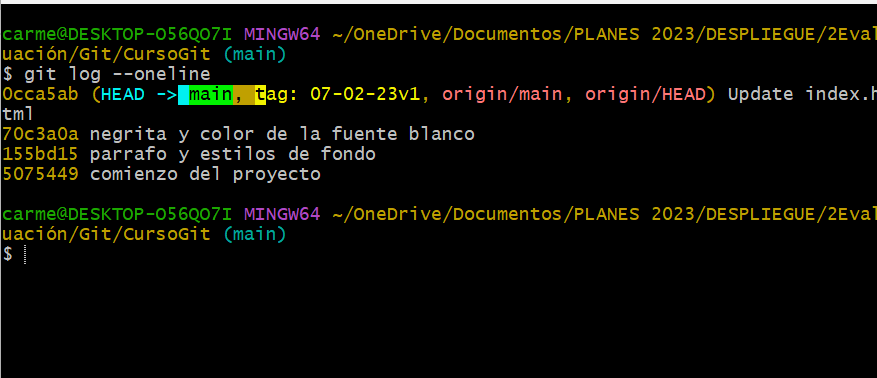
Descubres que durante la vida de desarrollo del proyecto has ido hacia atrás varias veces, además en un proyecto en un determinado momento del proyecto se hacen pruebas.

En nuestro proyecto, aunque es muy sencillo. Puede ser que una persona se encargue en el index otro en el css(usabilidad, navegabilidad y experta en CSS) y otra persona en javascript( experta en darle interactividad a la página web js)tres archivos diferentes que trabajan en el mismo proyecto.

Aquí es donde viene las ramas. Tenemos la rama Master o principal y las ramas copias que se han podido crear adicionalmente.

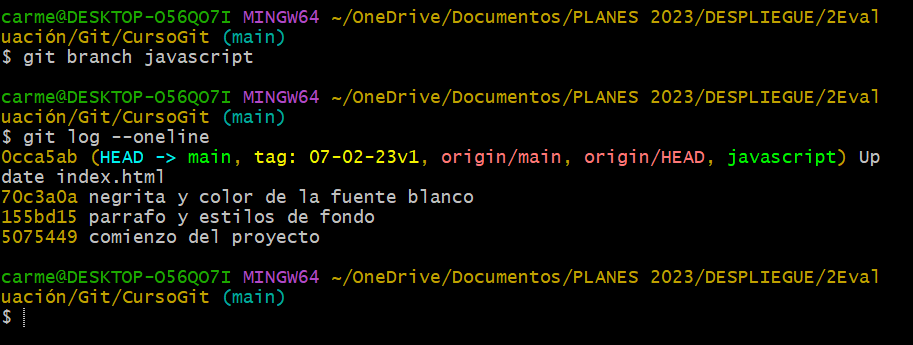


Problema: conflicto de concurrencia, modificando lo mismo del mismo archivo en las dos ramas, si trabajamos con tres o cuatro ramas aún se pone más compleja.

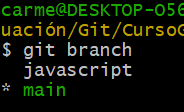


Main nos indica que estamos en la rama main o master

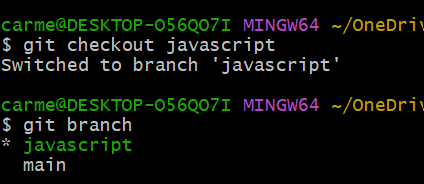
Voy a empezar a modificar el archivo js ( lo suyo es crear una rama para hacer pruebas)

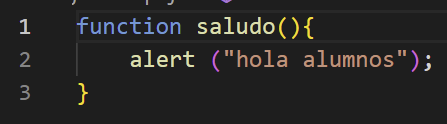


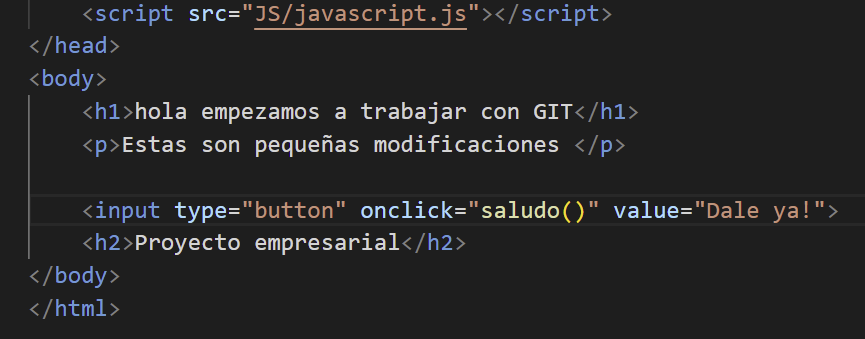
La imagen me indica que tengo otra rama javascript

Vemos las ramas del proyecto y en que rama me encuentro (está en verde y en \*)

Para movernos en javascript

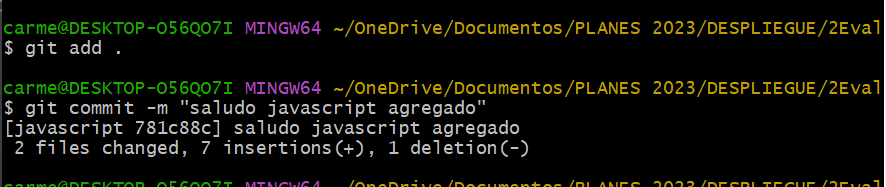
muevo mi flujo de trabajo temporal

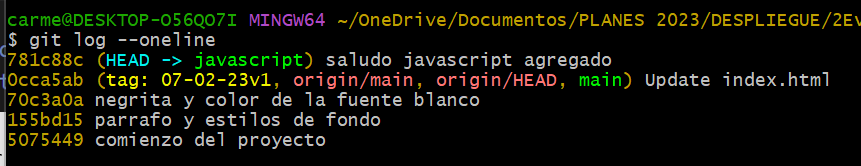
Agrego en js esta función linko en mi html y creo un botón que llame a la función.



Agregar las modificaciones en state área con el comando git add .

Y ahora hago un commit





1. TEMA 6: VISUAL STUDIO CODE CON GIT. RAMAS O BRANCHES II
2. TEMA 7: VISUAL STUDIO CODE CON GIT. RAMAS O BRANCHES III
3. TEMA 8: VISUAL STUDIO CODE CON GITHUB. SUBIENDO SINCRONIZANDO PROYECTO
4. TEMA 9: VISUAL STUDIO CODE CON GITHUB II. RAMAS
5. TEMA 10: FORK
6. TEMA 11: ELCLIPSE CON GIT Y GITHUB