UT3 Actividad 03 – Comandos GIT

Contenido

[1 Cambiar el nombre de la rama Master a Main 2](#_Toc151571417)

[1.1 Cambiar el nombre de una rama 2](#_Toc151571418)

[2 Archivo Readme.md y comando log 3](#_Toc151571419)

[2.1 Ver los commits realizados 4](#_Toc151571420)

[3 Adds y commits “visuales” con VSCode 4](#_Toc151571421)

[4 Seguimiento “stagging” 5](#_Toc151571422)

[4.1 Seguimiento de Carpetas vacías 5](#_Toc151571423)

[4.2 Archivo .gitkeep 7](#_Toc151571424)

[4.3 Añadir una carpeta y todo su contenido al seguimiento “stagging” 7](#_Toc151571425)

**A continuación, completa este manual con tus capturas tras la ejecución de los comandos, sustituyendo cuando sea necesario las imágenes por las tuyas.**

# Cambiar el nombre de la rama Master a Main

La rama principal de seguimiento del codigo fuente se denomina rama Master. Podemos crear tantas ramas “branch” como queramos para realizar cambios en el código fuente.

Por ejemplo, en un proyecto podríamos tener la necesidad de incorporar 2 funcionalidades diferentes, por ejemplo el login y una gestión de altas de usuario, para ello podríamos crear 2 ramas a partir de la master, una llamada “desarrolloLogin” y otra llamada “desarrolloAltasUsuarios”. Una vez finalizados y probados los desarrollos se integrarían sobre la rama principal.

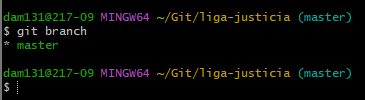
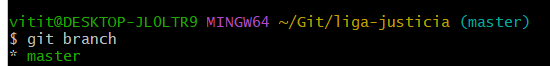
Por lo tanto, una **rama de desarrollo** (“Git Branch”) es una bifurcación del estado del código que crea un nuevo camino para la evolución del mismo.

En el prompt de gitbash aparece entre paréntesis el nombre de la rama en la que estamos trabajando,



Además el comando git branch nos indica en que rama estamos trabajando:

git branch

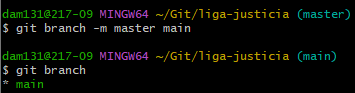


## Cambiar el nombre de una rama

Modifica el nombre de la rama “master” al nombre “main”

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

git branch -m master main



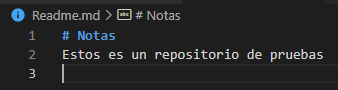
La opción -m indica que se va a cambiar de nombre a una rama.

# Archivo Readme.md y comando log

Desde VSCode crea un archivo nuevo en nuestro proyecto, al que llamaremos Readme.md:

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:



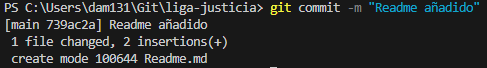
Le damos seguimiento **“stage”** al archivo:

git add Readme.md



Hacemos commit:

git commit -m “Readme añadido”

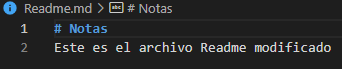


Ahora vamos a probar que esto funciona. Borra el archivo README.md desde VSCode. Como hemos hecho commit previamente podemos recuperarlo con git checkout:

git checkout -- .



Modificamos el archivo Readme:



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

git commit -am “Readme modificado”

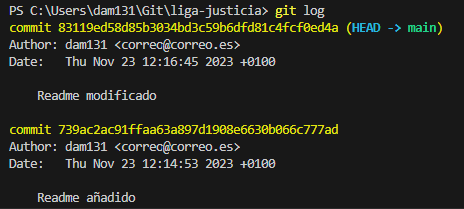


La opción -a indica que el commit tenga en cuenta todo tipo de ficheros, modificados y nuevos siempre y cuando les estemos dando seguimiento a los archivos, pero si estuviera marcado con la U de “untracked” es decir sin seguimiento no nos permitiría realizar el commit.

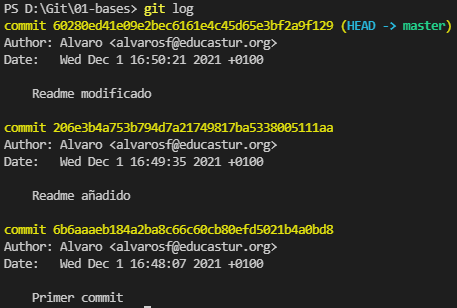
## Ver los commits realizados

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

git log



Vemos los tres commits que hemos hecho hasta ahora. La etiqueta head indica la última versión del repositorio:



Identificador (hash) del commit

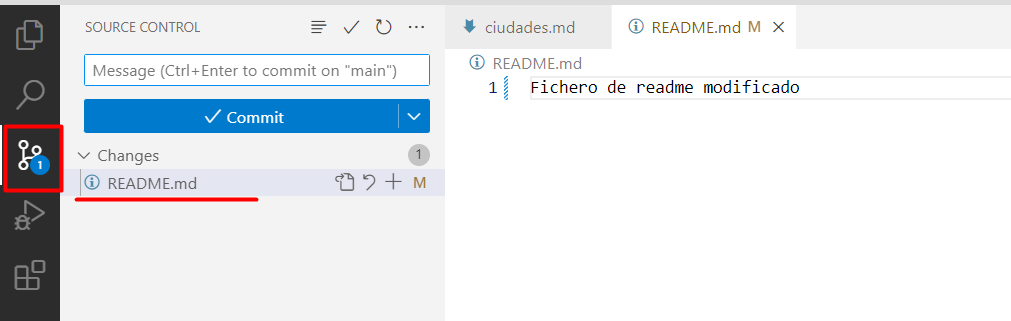
Autor del commit

Fecha y hora del commit

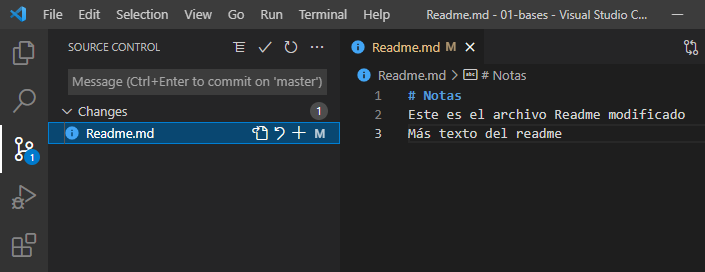
Texto identificativo del commit

# Adds y commits “visuales” con VSCode

En Visual studio code tenemos un apartado más visual para visualizar las operaciones de git:

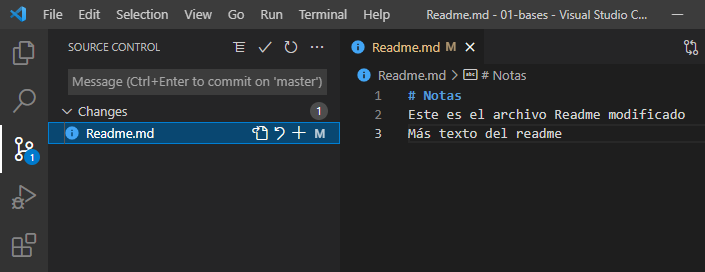


Con este icono abrimos el archivo:



Con este otro deshacemos los cambios realizados en el archivo:

Para añadir los cambios al escenario de seguimiento “Stage” usamos este botón (git add):



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.

# Seguimiento “stagging”

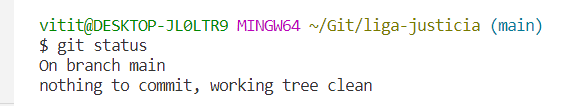
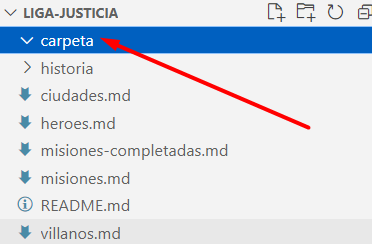
Como ya conocemos, en la nomenclatura de Git añadir al seguimiento los ficheros “stagging” es incorporar al escenario “stage” los archivos que deseemos.

## Seguimiento de Carpetas vacías

Git no hace seguimiento a las carpetas vacías.

Crea una carpeta nueva para verlo.

Si hacemos ahora git status:

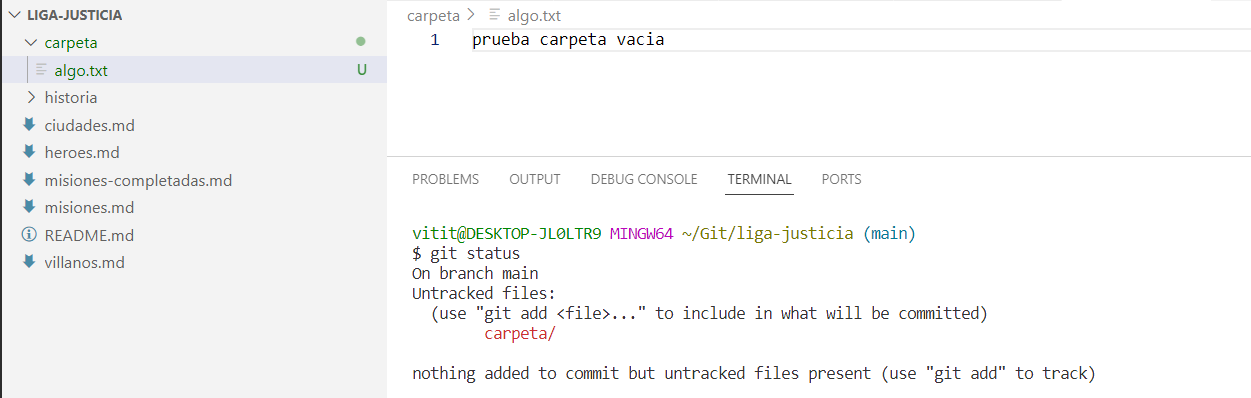


Vemos que git ni la muestra en la lista de archivos de los que no se hace seguimiento.

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:



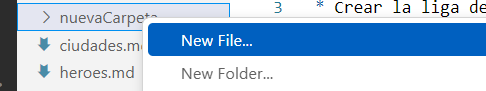
## Archivo .gitkeep

Cuando queramos añadir al menos un archivo a una carpeta vacía para que git la tenga en cuenta una convención que se suele utilizar por muchas personas es la de crear dentro de cualquier carpeta vacía un archivo vacío llamado .gitkeep

Renombramos “nueva carpeta” como “nuevacarpeta”

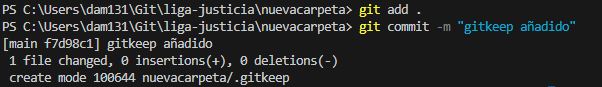


Añadimos con el editor el nuevo fichero



Hacemos add y commit:

git commit -m “.gitkeep añadido”



Si borramos la carpeta, podemos ver como la restauramos con:

git checkout -- .



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

## Añadir una carpeta y todo su contenido al seguimiento “stagging”

Siempre que queramos añadir todo el contenido del directorio al seguimiento lo más fácil es usar git add .

Sin embargo puede que queremos añadir por ejemplo todo el contenido de una carpeta concreta (tanto archivos como directorios que contiene) .

Crea una carpeta “java” y dentro 2 ficheros con extensión .java.

Haz el stagging de la carpeta y sus ficheros

git add java\



Y haz el commit con el mensaje “Añadidos directorio y ficheros java”:

git commit -m “Añadidos directorio y ficheros java”

