Actividad 03-01. Trabajar con GIT

Git se ha convertido en el estándar mundial para el control de versiones. Entonces, ¿qué es exactamente?

Git es un **sistema de control de versiones distribuido**, lo que significa que *un clon local del proyecto es un repositorio de control de versiones completo*. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad.

Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan su copia del repositorio con la copia en el servidor.

Este paradigma es distinto del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones.

La flexibilidad y popularidad de **Git** hacen que sea una excelente opción para cualquier equipo. Muchos desarrolladores y graduados universitarios ya saben cómo usar Git.

La comunidad de usuarios de Git ha creado recursos para entrenar a desarrolladores y la popularidad de Git facilita la ayuda cuando sea necesario. Casi todos los entornos de desarrollo tienen compatibilidad con Git y las herramientas de línea de comandos de Git implementadas en cada sistema operativo principal.

Ventajas de Git

Las ventajas de Git son muchas:

Desarrollo simultáneo

Todos tienen su propia copia local de código y pueden trabajar simultáneamente en sus propias ramas. Git funciona sin conexión, ya que casi todas las operaciones son locales.

Versiones más rápidas

Las ramas permiten el desarrollo flexible y simultáneo. La rama principal contiene código estable y de alta calidad desde el que se publica. Las ramas de características contienen trabajo en curso, que se combinan en la rama principal tras la finalización. Al separar la rama de versión del desarrollo en curso, es más fácil administrar código estable y enviar actualizaciones más rápidamente.

Integración integrada

Debido a su popularidad, *Git se integra en la mayoría de las herramientas y productos*. Cada IDE principal tiene compatibilidad integrada con Git y muchas herramientas admiten la integración continua, la implementación continua, las pruebas automatizadas, el seguimiento de elementos de trabajo, las métricas y la integración de características de informes con Git. Esta integración simplifica el flujo de trabajo diario.

Soporte técnico sólido de la comunidad

Git es de código abierto y se ha convertido en el estándar de facto para el control de versiones. No hay escasez de herramientas y recursos disponibles para que los equipos aprovechen. El volumen de compatibilidad de la comunidad con Git en comparación con otros sistemas de control de versiones facilita la ayuda cuando sea necesario.

Git funciona con cualquier equipo

El uso de Git con una herramienta de administración de código fuente aumenta la productividad de un equipo fomentando la colaboración, aplicando directivas, automatizando procesos y mejorando la visibilidad y la rastreabilidad del trabajo. El equipo puede establecerse en herramientas individuales para el control de versiones, el seguimiento de elementos de trabajo y la integración e implementación continuas. O bien, pueden elegir una solución como GitHub o Azure DevOps que admita todas estas tareas en un solo lugar.

Solicitudes de incorporación de cambios

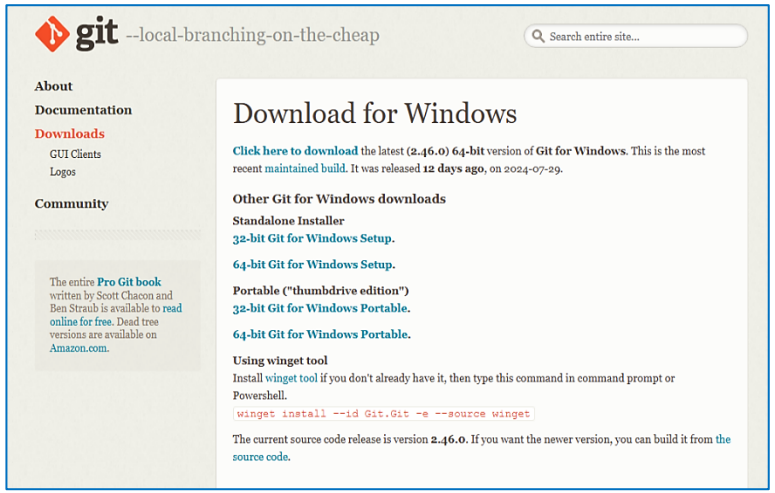
Use solicitudes de incorporación de cambios para analizar los cambios de código con el equipo antes de combinarlos en la rama principal. Las discusiones en las solicitudes de incorporación de cambios son valiosas para garantizar la calidad del código y aumentar el conocimiento en todo el equipo. Las plataformas como GitHub y Azure DevOps ofrecen una experiencia enriquecida de solicitudes de incorporación de cambios donde los desarrolladores pueden examinar los cambios de archivos, dejar comentarios, inspeccionar confirmaciones, ver compilaciones y votar para aprobar el código.

Instalación de Git (Windows)

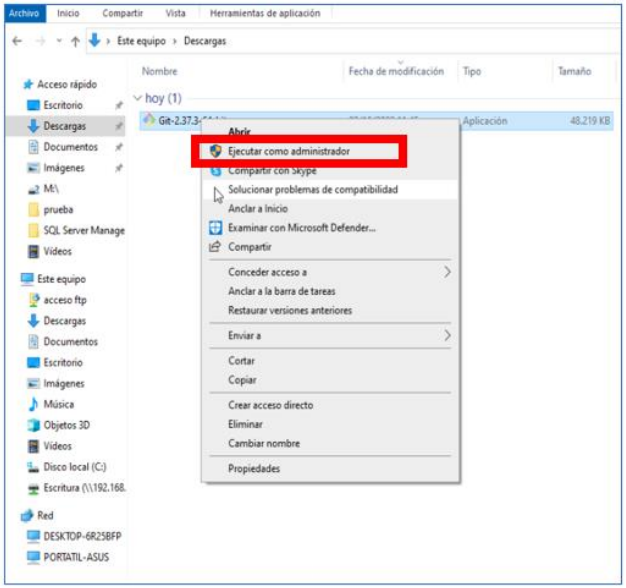
1. Descarga e instala Git

Accede a Git en el siguiente enlace: [Git:]

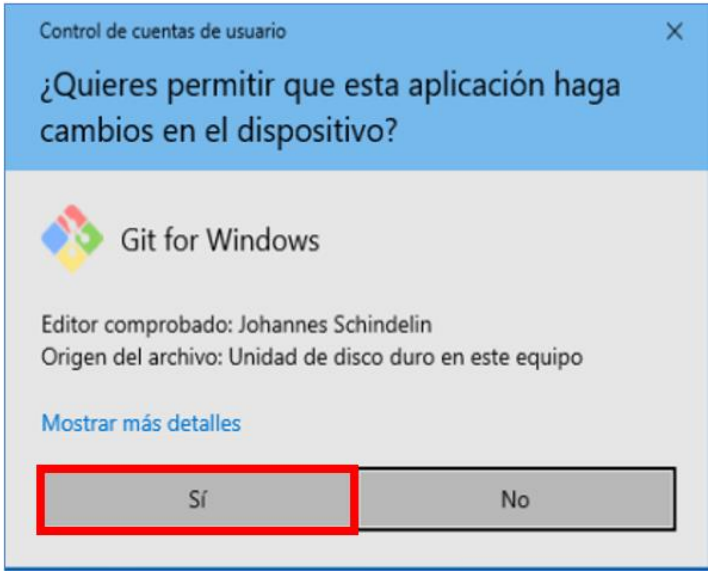
Descarga el programa de instalación de Windows:



Ejecutar el instalador (En modo administrador):

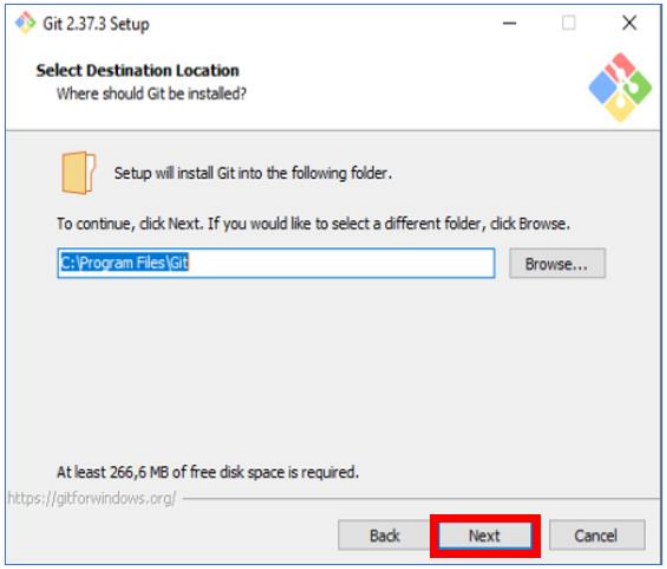


Comprobación del Sistema operativo del fabricante:

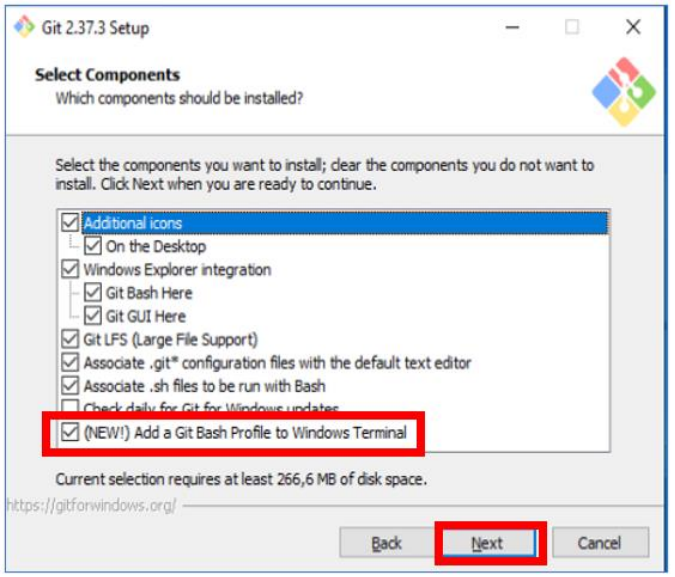


Aceptación de la licencia de uso:

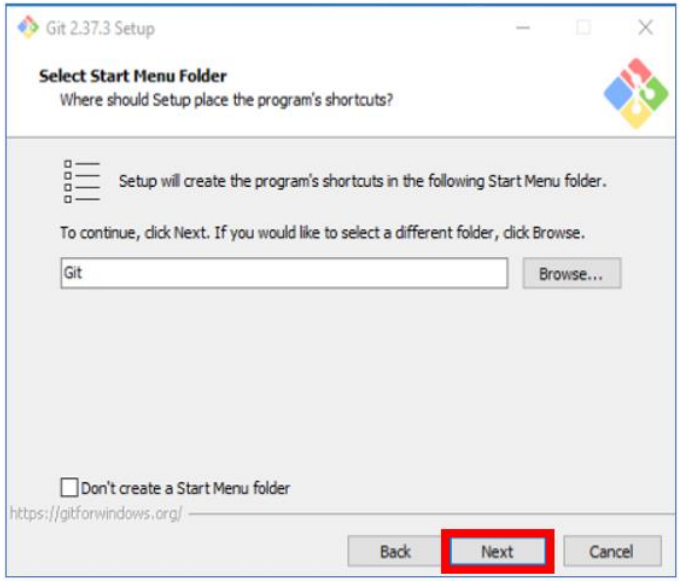
Elegir el lugar de instalación del programa:



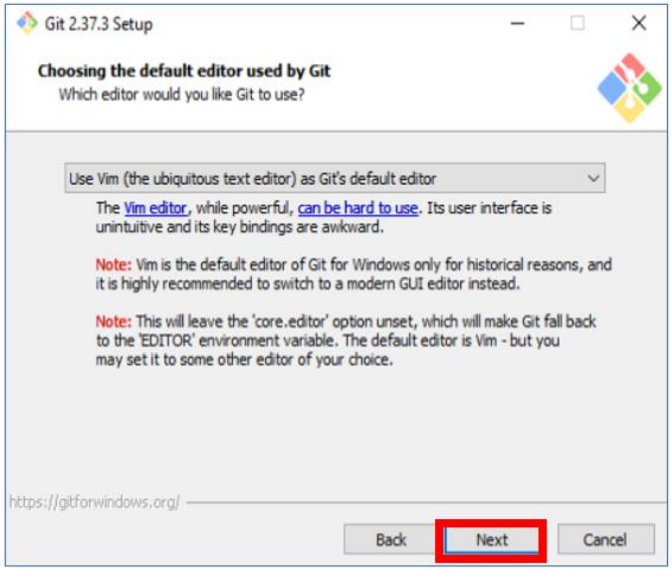
Selección de componentes:



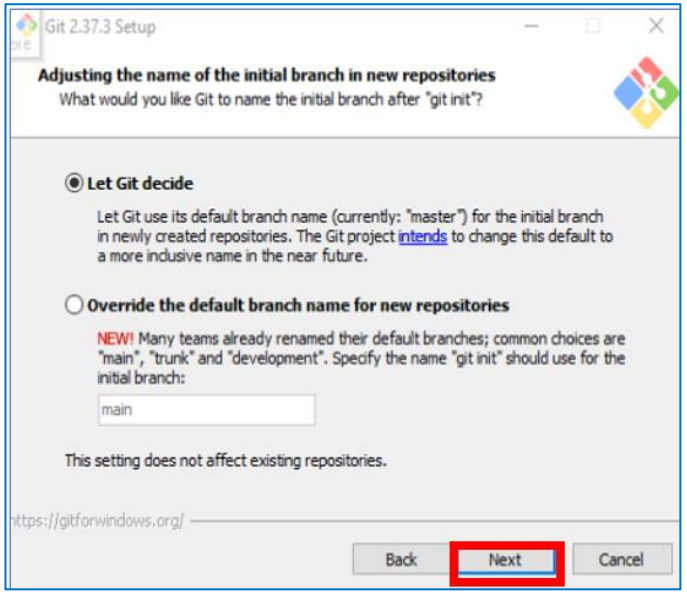
Selección carpeta de menú de inicio:

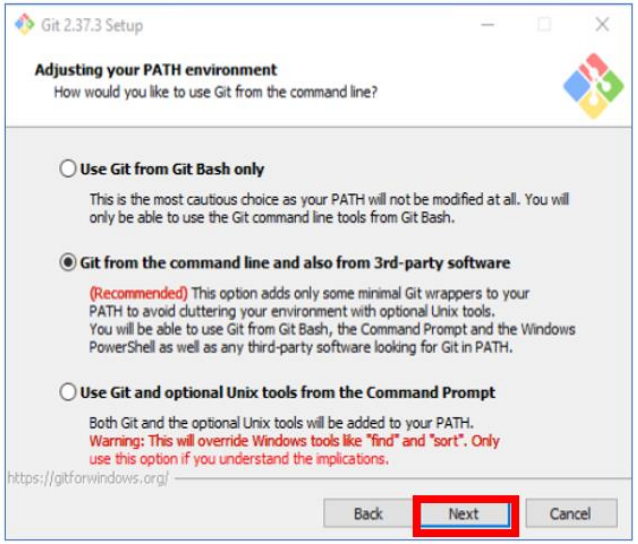


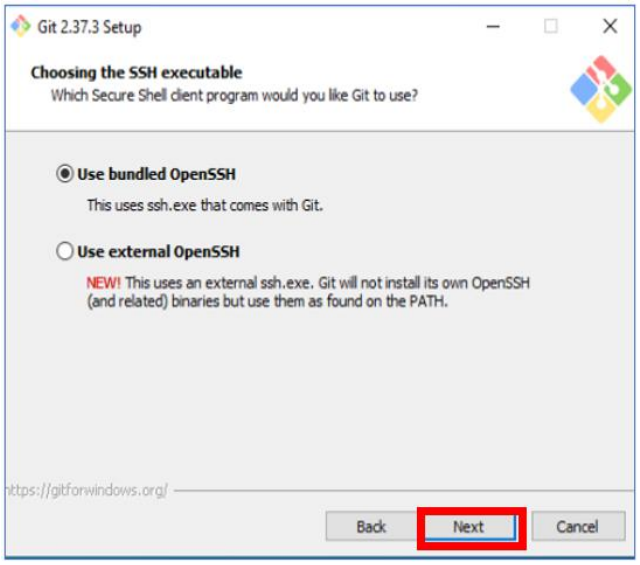
Selección editor por defecto:

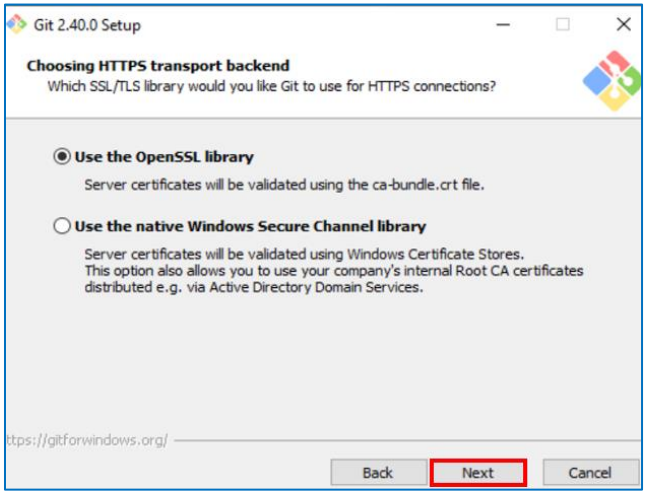


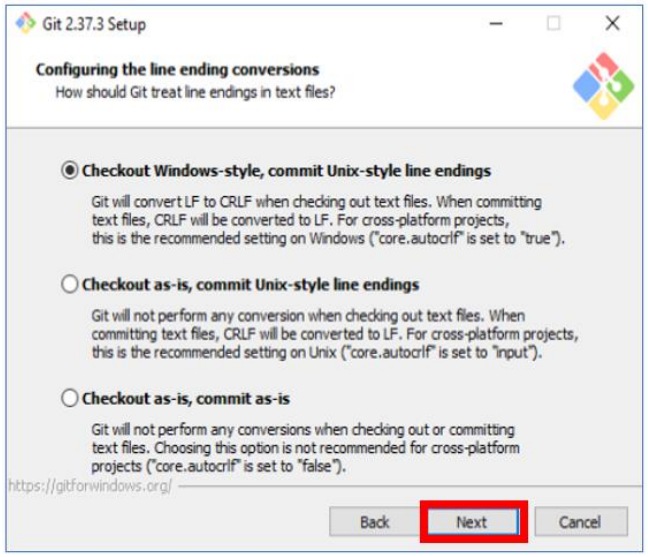
Selección de varias opciones:

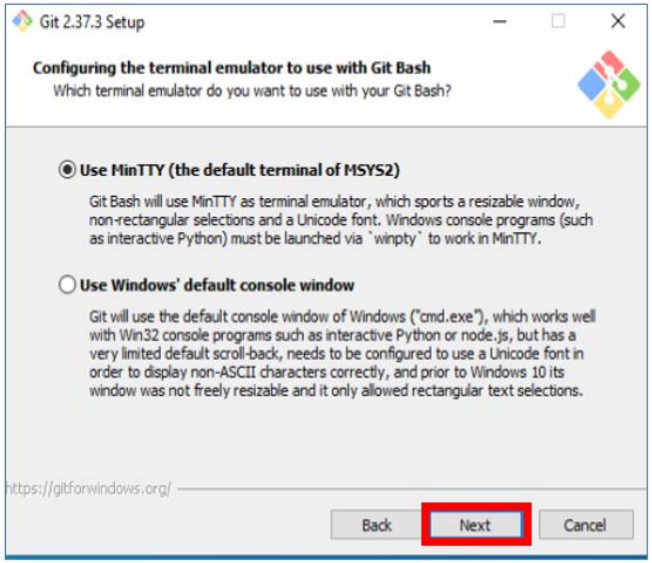


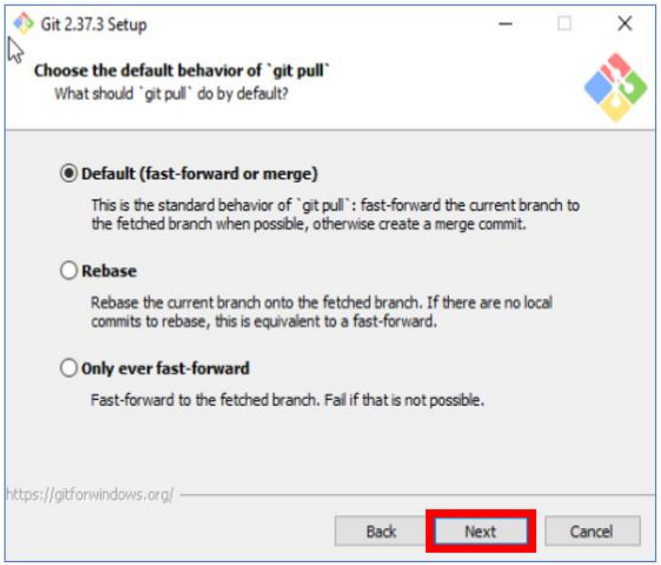

Git 2.37.3 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.
Git from the command line and also from 3rd-party software
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools.
You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.
Use Git and optional Unix tools from the Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.
https://gitforwindows.org/
Back
Next
Cancel

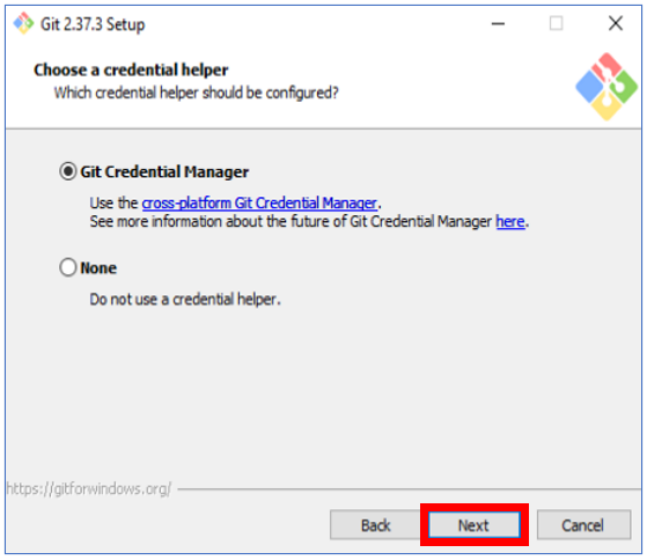

Git 2.37.3 Setup
Choosing the SSH executable
Which Secure Shell client program would you like Git to use?
Use bundled OpenSSH
This uses ssh.exe that comes with Git.
Use external OpenSSH
NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.
https://gitforwindows.org/
Back
Next
Cancel

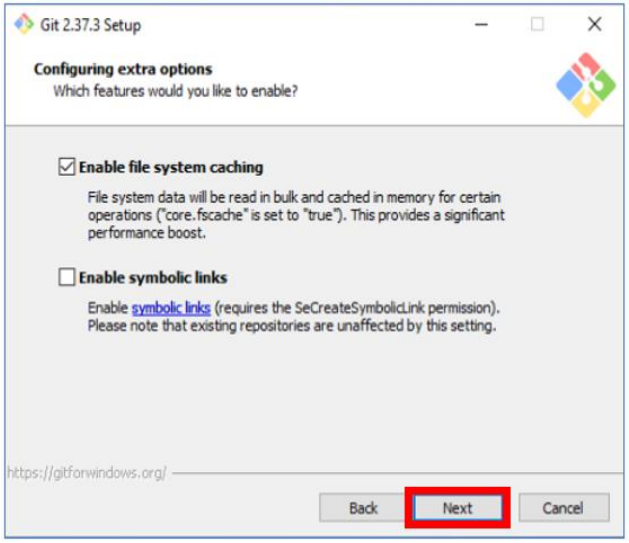

Git 2.40.0 Setup
Choosing HTTPS transport backend
Which SSL/TLS library would you like Git to use for HTTPS connections?
Use the OpenSSL library
Server certificates will be validated using the ca-bundle.crt file.
Use the native Windows Secure Channel library
Server certificates will be validated using Windows Certificate Stores.
This option also allows you to use your company's internal Root CA certificates
distributed e.g. via Active Directory Domain Services.
ttps://gitforwindows.org/
Back
Next
Cancel

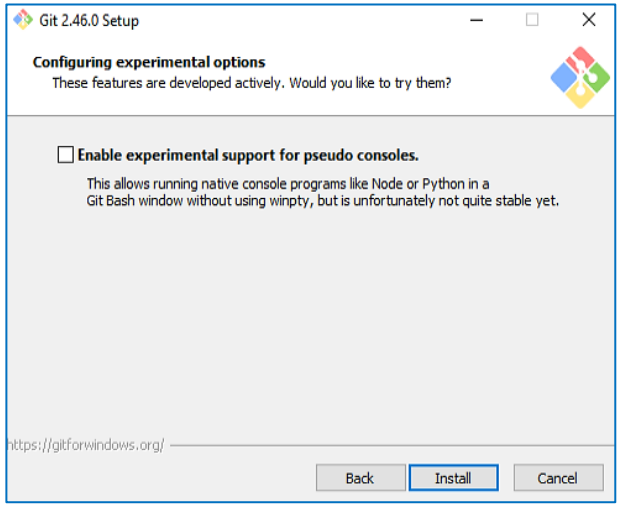

Git 2.37.3 Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing
text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When
committing text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing
text files. Choosing this option is not recommended for cross-platform
projects ("core.autocrlf" is set to "false").
https://gitforwindows.org/
Back
Next
Cancel


Git 2.37.3 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.
https://gitforwindows.org/
Back
Next
Cancel

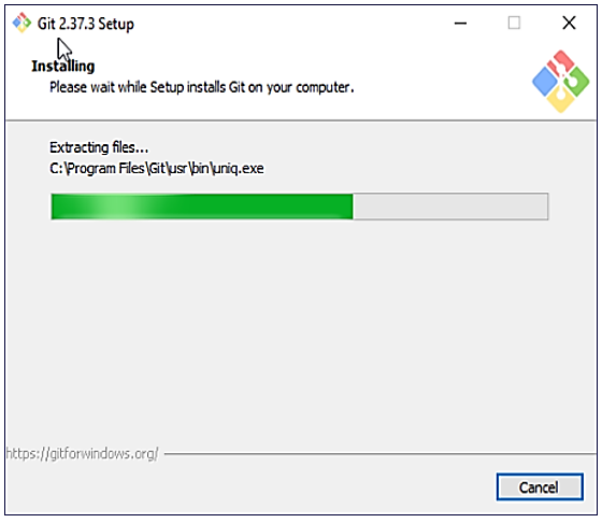

Git 2.37.3 Setup
Choose the default behavior of `git pull`
What should `git pull` do by default?
Default (fast-forward or merge)
This is the standard behavior of `git pull`: fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.
Rebase
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.
Only ever fast-forward
Fast-forward to the fetched branch. Fail if that is not possible.
https://gitforwindows.org/
Back
Next
Cancel


Git 2.37.3 Setup
Choose a credential helper
Which credential helper should be configured?
Git Credential Manager
Use the cross-platform Git Credential Manager.
See more information about the future of Git Credential Manager here.
None
Do not use a credential helper.
https://gitforwindows.org/
Back
Next
Cancel


Git 2.37.3 Setup
Configuring extra options
Which features would you like to enable?
Enable file system caching
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.
Enable symbolic links
Enable symbolic links (requires the SeCreateSymbolicLink permission).
Please note that existing repositories are unaffected by this setting.
https://gitforwindows.org/
Back
Next
Cancel



Comienza la instalación:



1.Descarga e instala Git (Linux)

Veamos cómo se realiza la instalación de Git en Linux Ubuntu:

Primero, se actualiza el sistema operativo:

```
sudo apt update
sudo apt upgrade
```

A continuación, se instala Git:

```
sudo apt install git
```

2. Crear un repositorio

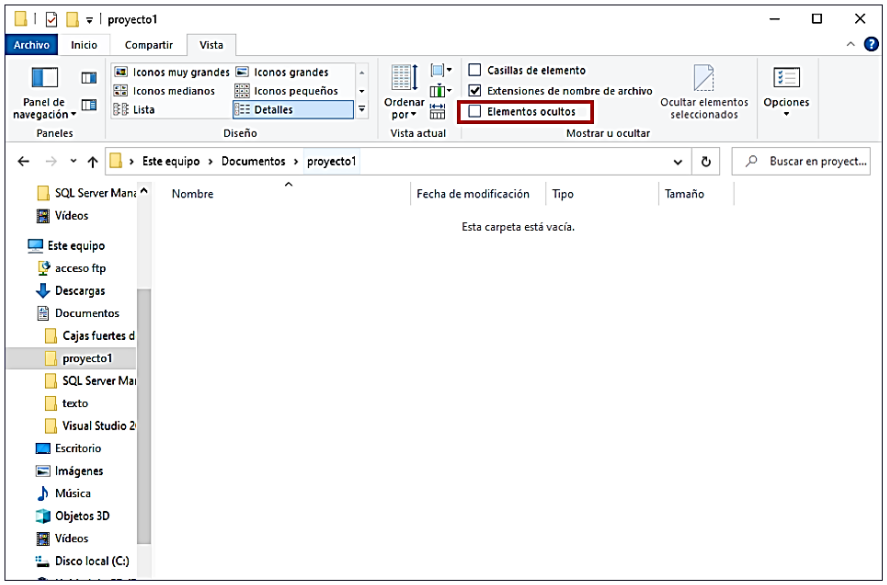
Seguir los siguientes pasos:

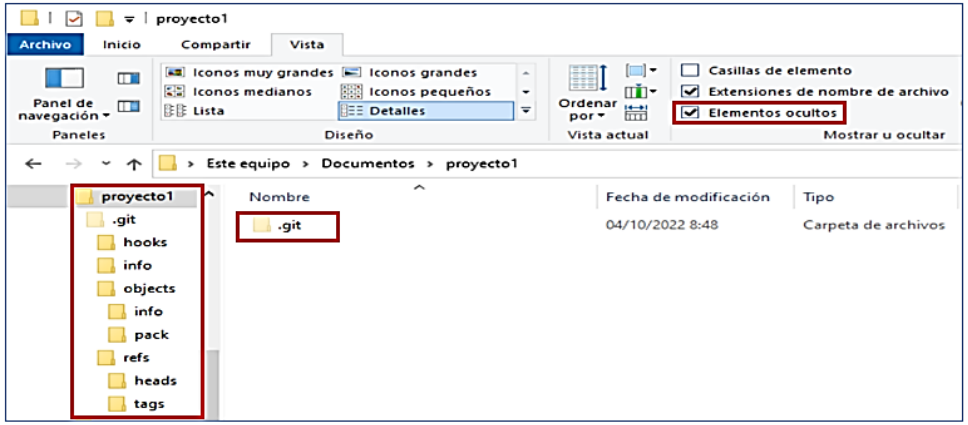
- Iniciar Git Bash

- Crear directorio (proyecto1)
- Moverse al directorio (proyecto1)
- Crear repositorio (**git init**)

```
docente@DESKTOP-6R25BFP MINGW64 ~
$ pwd
/c/Users/docente

docente@DESKTOP-6R25BFP MINGW64 ~
$ cd Documents

docente@DESKTOP-6R25BFP MINGW64 ~/Documents
$ mkdir proyecto1

docente@DESKTOP-6R25BFP MINGW64 ~/Documents
$ cd proyecto1

docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1
$ pwd
/c/Users/docente/Documents/proyecto1

docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1
$ git init
Initialized empty Git repository in C:/Users/docente/Documents/proyecto1/.git/

docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$
```

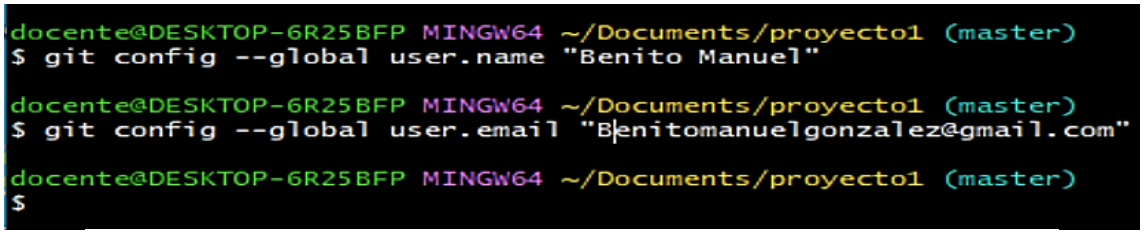
Si visualizamos la carpeta proyecto, aparentemente, no hay nada. Pero, si visualizamos los archivos ocultos, podremos ver que se ha creado la estructura de Git para ese repositorio:

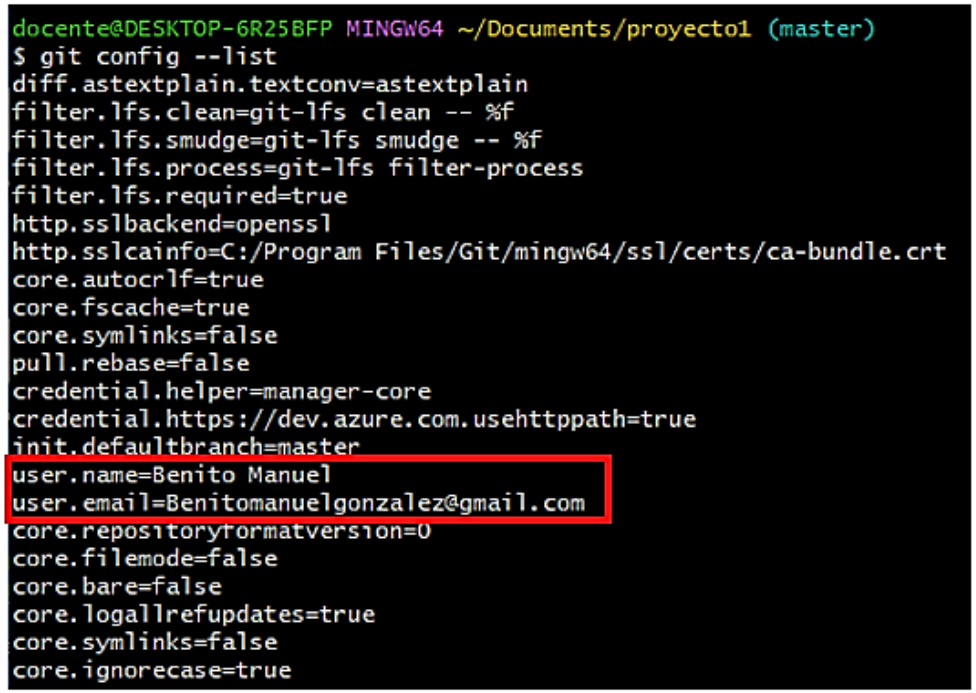




Antes de comenzar a usar el repositorio, vamos a añadir nuestros datos en la configuración de Git. De esta manera, los cambios que realice se registrarán con nuestros datos:

```
git config --global user.name "Mi nombre"
git config --global user.email cuentacorreo@gmail.com
git config --list
```



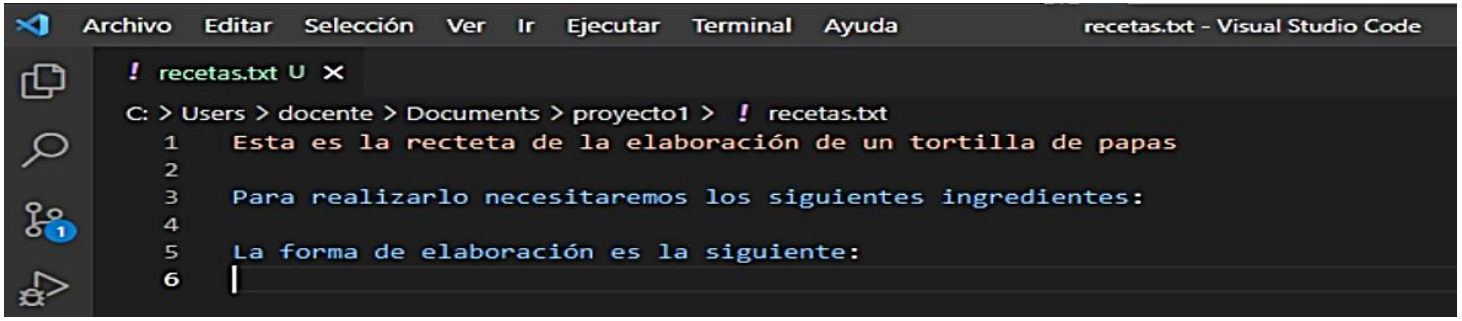


2. Crea y modifica documento en Git

Vamos codea crear un archivo nuevo dentro de Git. Para ello, utilizamos **Visual Studio Code**:

```
docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$ code
```

Y lo guardamos (recetas.txt) en la carpeta en la que hemos creado el repositorio:



Ahora, si hacemos ls, nos mostrará el archivo. Si escribimos el comando **git status**, nos indica que hay un archivo nuevo, pero no está guardardo en Git:

```
docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$ ls
recetas.txt

docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        recetas.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Si queremos añadirlo, usamos el comando **git add:**

```
docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$ git add recetas.txt

docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   recetas.txt
```

Ahora se ha registrado el cambio, pero todavía no se ha registrado en la base de datos de Git. Para ello, se hace **commit** (Es recomendable poner un comentario):

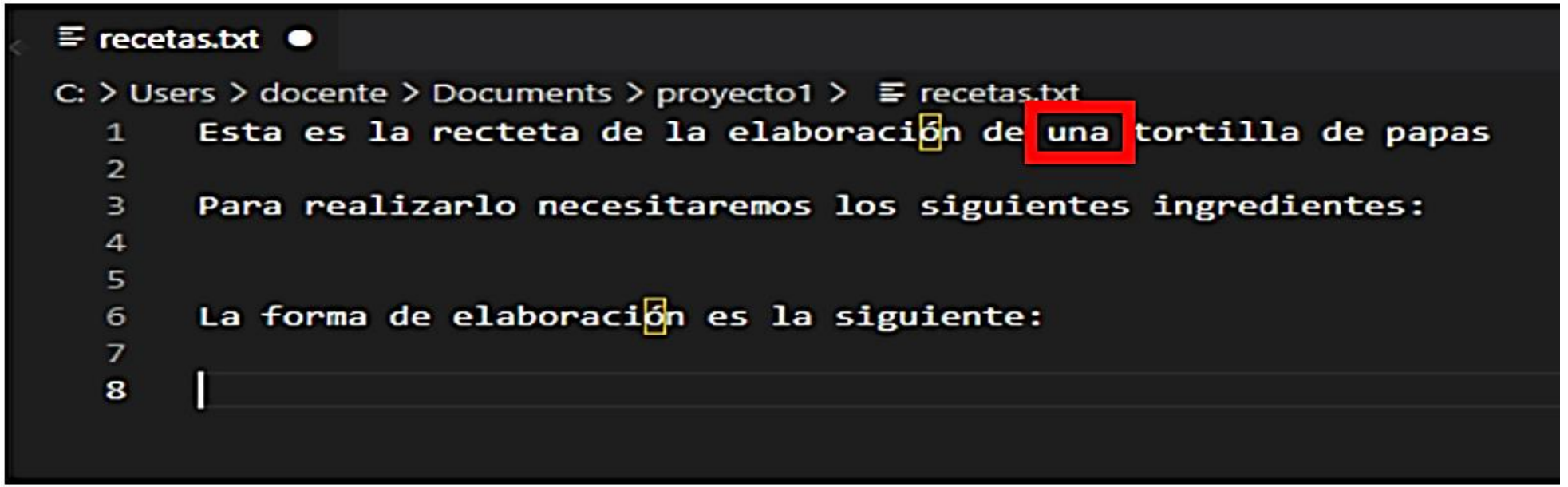
```
docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$ git commit -m "He creado el archivo recetas.txt y escrito algo de texto"
[master (root-commit) 2a3651b] He creado el archivo recetas.txt y escrito algo
 de texto
 1 file changed, 7 insertions(+)
 create mode 100644 recetas.txt
```

Comprobamos que los cambios se han registrado:

```
docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$ Git status
On branch master
nothing to commit, working tree clean
```

Vamos a hacer un cambio en el documento, para ello, utilizamos **Code**:

```
docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$ code recetas.txt
```



Si ahora hacemos **git status**, nos indicará que el archivo ha sido modificado:

```
$ 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:   recetas.txt

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

Hacemos de nuevo un **Git add,** para indicar ese nuevo cambio:

```
docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$ git add recetas.txt

docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   recetas.txt
```

Y hacemos un **commit**, para registrarlo en la base de datos:

```
$ git commit -m "Cambio en linea 1 de archivo recetas.txt"
[master f1b33c3] Cambio en linea 1 de archivo recetas.txt
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Añadimos mas texto al documento, hacemos un **add,** un **commit** y un **log**:

recetas.txt

proyecto1 > recetas.txt

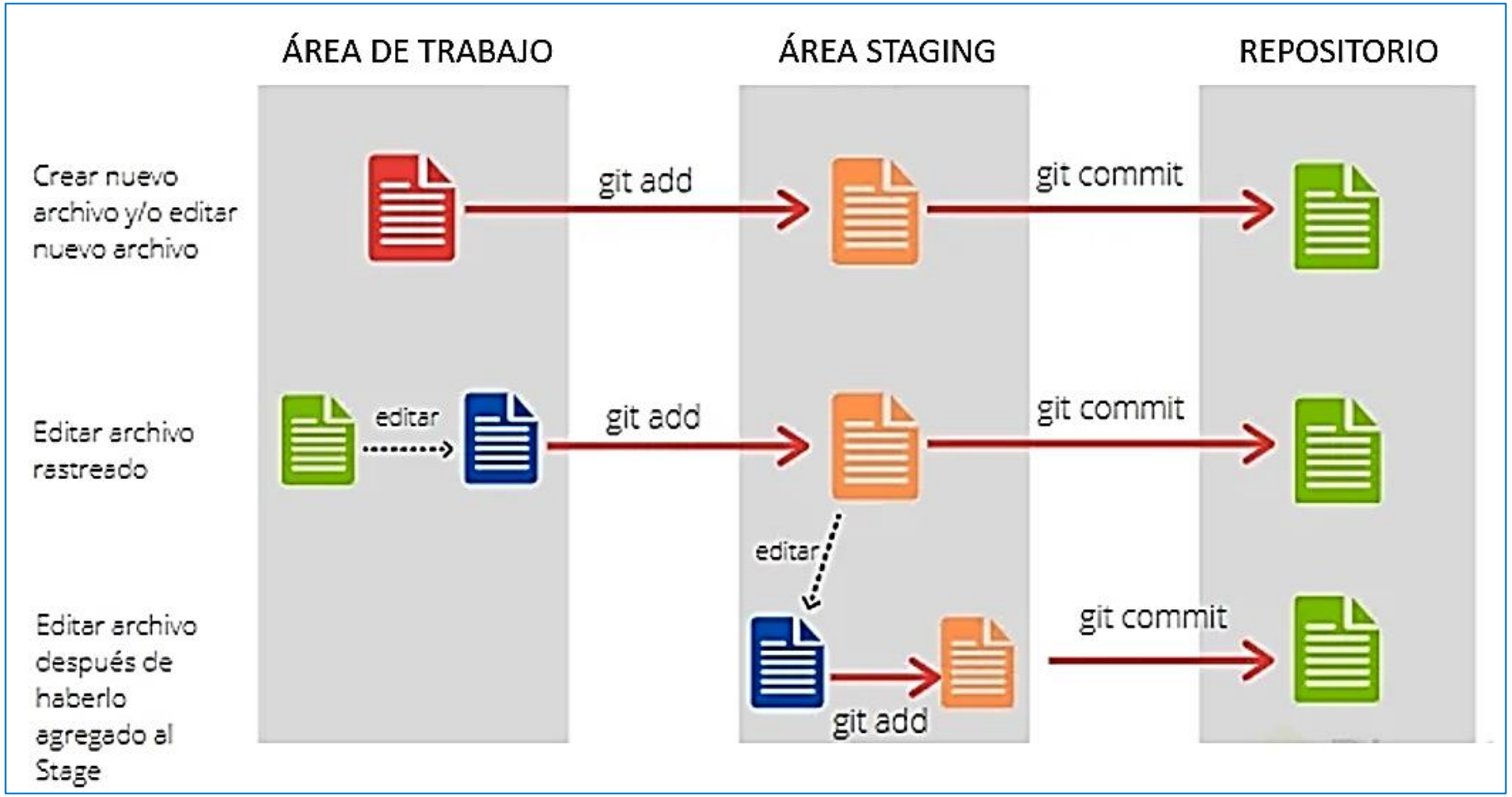
```
1   Esta es la recteta de la elaboración de una tortilla de papas
2
3   Para realizarlo necesitaremos los siguientes ingredientes:
4   1/2 litro de aceite de oliva (virgen extra)
5   Sal
6   8 huevos
7   1 cebolla
8   1 pimiento
9
10  La forma de elaboración es la siguiente:
11  Se pelan y se lavan las papas (se calcula una patata grande por persona). Se cortan en láminas no muy
12  Se fríen las papas en aceite abundante. Para hacer la tortilla es muy importante que las papas estén b
13  En un recipiente grande (por ejemplo una ensaladera) se baten los huevos (se calcula un huevo por pers
14  Cuando las papas ya están blandas, se sacan de la sartén y se escurren de aceite. Entonces se mezclan
15  Mientras la mezcla de los huevos y las papas reposar, se pica la cebolla en trozos muy pequeños y se f
16  En una sartén un poco profunda se echan 3 (tres) grandes cucharadas de aceite. Cuando esta caliente, s
17  Después de un par de minutos en la sartén, se da la vuelta a la tortilla (¡cuidado aquí, ¡ hacerlo en
18
19
20
```

Vemos todos los cambio del archivo recetas.txt.

```
$ git add recetas.txt

docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$ git commit -m "Se añaden los ingredientes y la preparación a recetas.txt" recetas.txt
[master eb9049c] Se añaden los ingredientes y la preparación a recetas.txt
 1 file changed, 13 insertions(+), 1 deletion(-)

docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$ git log recetas.txt
commit eb9049c5a81a971bf0390aaeee50128e97e83d83 (HEAD -> master)
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 12:28:28 2022 +0200

    Se añaden los ingredientes y la preparación a recetas.txt

commit f1b33c3c4d8c869ab459f6c984099cd96dd0fe5f
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 09:23:22 2022 +0200

    Cambio en linea 1 de archivo recetas.txt

commit 2a3651b188f83cdf0c40d5089e30027b192250fc
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 08:43:29 2022 +0200

    He creado el archivo recetas.txt y escrito algo de texto

docente@DESKTOP-6R25BFP MINGW64 ~/Documents/proyecto1 (master)
$
```

3. Funcionamiento de Git

Ahora, que hemos visto como añadir un documento al repositorio y luego modificarlo, veamos el esquema de funcionamiento:



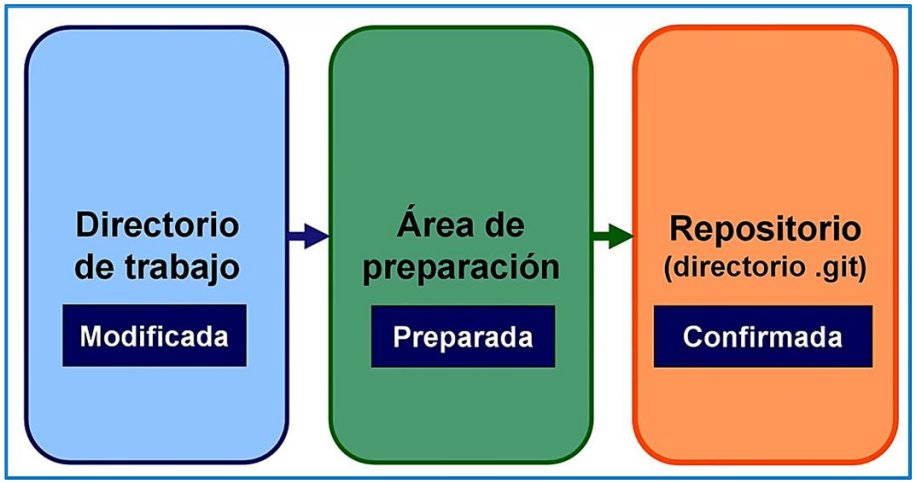
Cuando trabajamos con un documento, en la carpeta del repositorio, se dice que está *sin rastrear* (untracked). Cuando se hace un **git add**, el archivo pasa a estar rastreado (tracked), pasando su estado al área Staging.

Todavía no están agregados al repositorio, para hacerlo, hay que realizar un **git commit**. De esta forma, los cambios realizados ya están registrados en el repositorio (master).

Si se modifica un archivo, no se ha registrado hasta que hagamos un **git add**, de forma que ese cambio se refleja en el área Staging. Por último, para que quede registrado en el repositorio hay que hacer un **git commit**.

Por tanto, dentro de un repositorio GIT, un archivo puede estar en tres estados:

- **Modificado (modified)**, cuando el archivo se encuentra en el **Área De Trabajo**.
- **Preparado (staged)**, cuando el archivo se encuentra en el Área **Staging.**
- **Confirmado (commited)**, cuando el archivo se encuentra en el **Repositorio.**



En el documento recetas, vemos que hay una palabra mal escrita y la modificamos:

proyecto1 > recetas.txt

```
1   Esta es la recteta de la elaboración de una tortilla de papas
2   t
3   Para realizarlo necesitaremos los siguientes ingredientes:
4   1/2 litro de aceite de oliva (virgen extra)
5   Sal
6   8 huevos
7   1 cebolla
8   1 pimiento
9
10  La forma de elaboración es la siguiente:
11  Se pelan y se lavan las papas (se calcula una patata grande por persona). Se cortan en láminas no muy
12  Se fríen las papas en aceite abundante. Para hacer la tortilla es muy importante que las papas estén b
13  En un recipiente grande (por ejemplo una ensaladera) se baten los huevos (se calcula un huevo por pers
14  Cuando las papas ya están blandas, se sacan de la sartén y se escurren de aceite. Entonces se mezclan
15  Mientras la mezcla de los huevos y las papas reposar, se pica la cebolla en trozos muy pequeños y se f
16  En una sartén un poco profunda se echan 3 (tres) grandes cucharadas de aceite. Cuando esta caliente, s
17  Después de un par de minutos en la sartén, se da la vuelta a la tortilla (¡cuidado aquí, ¡ hacerlo en
18
```

proyecto1 > recetas.txt

```
1   Esta es la receta de la elaboración de una tortilla de papas
2   t
3   Para realizarlo necesitaremos los siguientes ingredientes:
4   1/2 litro de aceite de oliva (virgen extra)
5   Sal
6   8 huevos
7   1 cebolla
8   1 pimiento
9
10  La forma de elaboración es la siguiente:
11  Se pelan y se lavan las papas (se calcula una patata grande por persona). Se cortan en láminas no muy
12  Se fríen las papas en aceite abundante. Para hacer la tortilla es muy importante que las papas estén b
13  En un recipiente grande (por ejemplo una ensaladera) se baten los huevos (se calcula un huevo por pers
14  Cuando las papas ya están blandas, se sacan de la sartén y se escurren de aceite. Entonces se mezclan
15  Mientras la mezcla de los huevos y las papas reposar, se pica la cebolla en trozos muy pequeños y se f
16  En una sartén un poco profunda se echan 3 (tres) grandes cucharadas de aceite. Cuando esta caliente, s
17  Después de un par de minutos en la sartén, se da la vuelta a la tortilla (¡cuidado aquí, ¡ hacerlo en
18
```

El documento está modificado, pero esos cambios no se han añadido en el **área Staging**. Si hacemos un git status, vemos que se detecta que el archivo está modificado, pero no han sido añadidos los cambios:

```
$ 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:   recetas.txt

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

Si hacemos un **git log**, vemos que no se ha reflejado todavía el cambio:

```
$ git log recetas.txt
commit eb9049c5a81a971bf0390aaeee50128e97e83d83 (HEAD -> master)
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 12:28:28 2022 +0200

    Se añaden los ingredientes y la preparación a recetas.txt

commit f1b33c3c4d8c869ab459f6c984099cd96dd0fe5f
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 09:23:22 2022 +0200

    Cambio en linea 1 de archivo recetas.txt

commit 2a3651b188f83cdf0c40d5089e30027b192250fc
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 08:43:29 2022 +0200

    He creado el archivo recetas.txt y escrito algo de texto
```

Si hacemos un **git add**, se reflejarán los cambios:

```
pru@DESKTOP-BHSLPOA MINGW64 ~/Documents/proyecto1 (master)
$ git add recetas.txt

pru@DESKTOP-BHSLPOA MINGW64 ~/Documents/proyecto1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   recetas.txt
```

Ahora, los cambios están reflejados en el **área Staging**, pero todavía no están en el repositorio. Lo podemos comprobar haciendo un **git log**:

```
$ git log recetas.txt
commit eb9049c5a81a971bf0390aaeee50128e97e83d83 (HEAD -> master)
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 12:28:28 2022 +0200

    Se añaden los ingredientes y la preparación a recetas.txt

commit f1b33c3c4d8c869ab459f6c984099cd96dd0fe5f
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 09:23:22 2022 +0200

    Cambio en linea 1 de archivo recetas.txt

commit 2a3651b188f83cdf0c40d5089e30027b192250fc
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 08:43:29 2022 +0200

    He creado el archivo recetas.txt y escrito algo de texto
```

Para que se queden registrados en el repositorio, hay que hacer un **git commit**:

```
$ git commit -m "Se ha corregido la palabra receta"
[master 8854759] Se ha corregido la palabra receta
 1 file changed, 2 insertions(+), 2 deletions(-)
```

Ahora, si hacemos un **git log**, si se refleja el último cambio:

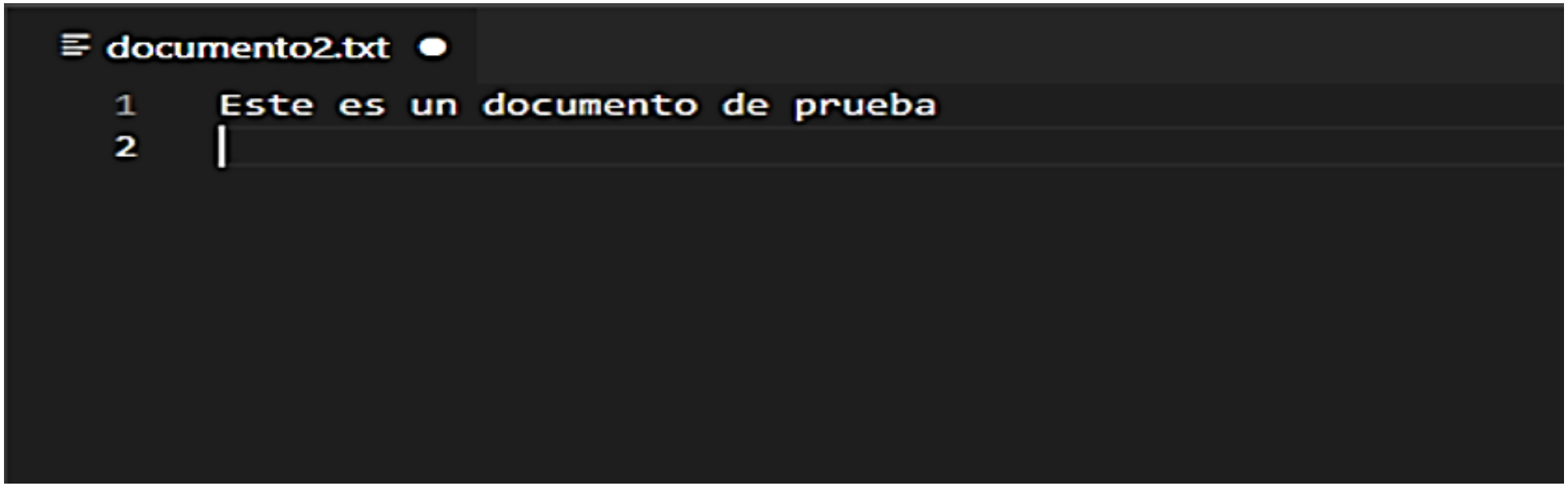
```
$ git log recetas.txt
commit 885475941414445e41eb2d03b7ebfa046fe1bc228 (HEAD -> master)
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Thu Oct 6 08:07:53 2022 +0200

    Se ha corregido la palabra receta

commit eb9049c5a81a971bf0390aaeee50128e97e83d83
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 12:28:28 2022 +0200

    Se añaden los ingredientes y la preparación a recetas.txt

commit f1b33c3c4d8c869ab459f6c984099cd96dd0fe5f
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 09:23:22 2022 +0200

    Cambio en linea 1 de archivo recetas.txt

commit 2a3651b188f83cdf0c40d5089e30027b192250fc
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 08:43:29 2022 +0200

    He creado el archivo recetas.txt y escrito algo de texto
```

Cada vez que realizamos un **commit**, estamos añadiendo una nueva versión del documento al **repositorio.** Git le añade el número de versión.

4. Eliminar documento del repositorio Git

Vamos a añadir un nuevo documento (doumento2) en la carpeta proyecto1:

```
$ code documento2.txt
```



Lo añadimos al repositorio:

```
$ git add documento2.txt

docente@DESKTOP-6R25BFP MINGW64 ~/documents/proyecto1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   documento2.txt


docente@DESKTOP-6R25BFP MINGW64 ~/documents/proyecto1 (master)
$ git commit -m "Añadido documento2.txt"
[master 6153740] Añadido documento2.txt
 1 file changed, 1 insertion(+)
 create mode 100644 documento2.txt

docente@DESKTOP-6R25BFP MINGW64 ~/documents/proyecto1 (master)
$ git log
commit 61537409a2b4c66aaa0d1cb498761d10e35d9bb3 (HEAD -> master)
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Thu Oct 6 08:53:28 2022 +0200

    Añadido documento2.txt

commit 885475941414445e41eb2d03b7ebfa046fe1bc228
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Thu Oct 6 08:07:53 2022 +0200

    Se ha corregido la palabra receta

commit eb9049c5a81a971bf0390aaeee50128e97e83d83
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 12:28:28 2022 +0200

    Se añaden los ingredientes y la preparación a recetas.txt

commit f1b33c3c4d8c869ab459f6c984099cd96dd0fe5f
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 09:23:22 2022 +0200

    Cambio en linea 1 de archivo recetas.txt

commit 2a3651b188f83cdf0c40d5089e30027b192250fc
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 08:43:29 2022 +0200

    He creado el archivo recetas.txt y escrito algo de texto
```

Si ahora eliminamos el documento:

```
$ rm documento2.txt

docente@DESKTOP-6R25BFP MINGW64 ~/documents/proyecto1 (master)
$ ls
recetas.txt
```

Para eliminarlo del repositorio, lo eliminamos del **área Staging** y luego hacemos **commit**:

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    documento2.txt

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

docente@DESKTOP-6R25BFP MINGW64 ~/documents/proyecto1 (master)
$ git rm documento2.txt
rm 'documento2.txt'

docente@DESKTOP-6R25BFP MINGW64 ~/documents/proyecto1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    documento2.txt


docente@DESKTOP-6R25BFP MINGW64 ~/documents/proyecto1 (master)
$ git commit -m "Eliminado documento2.txt"
[master 9e27ad7] Eliminado documento2.txt
 1 file changed, 1 deletion(-)
 delete mode 100644 documento2.txt
```

Si hacemos **git log**, vemos el registro de los cambios realizados:

```
$ git log
commit 9e27ad741e881b28aeea74c583a405048bf138cf (HEAD -> master)
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Thu Oct 6 08:58:14 2022 +0200

    Eliminado documento2.txt

commit 61537409a2b4c66aaa0d1cb498761d10e35d9bb3
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Thu Oct 6 08:53:28 2022 +0200

    Añadido documento2.txt

commit 885475941414445e41eb2d03b7ebfa046fe1bc228
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Thu Oct 6 08:07:53 2022 +0200

    Se ha corregido la palabra receta

commit eb9049c5a81a971bf0390aaeee50128e97e83d83
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 12:28:28 2022 +0200

    Se añaden los ingredientes y la preparación a recetas.txt

commit f1b33c3c4d8c869ab459f6c984099cd96dd0fe5f
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 09:23:22 2022 +0200

    Cambio en linea 1 de archivo recetas.txt

commit 2a3651b188f83cdf0c40d5089e30027b192250fc
Author: Benito Manuel <Benitomanuelgonzalez@gmail.com>
Date:   Wed Oct 5 08:43:29 2022 +0200

    He creado el archivo recetas.txt y escrito algo de texto
```

Ejemplos de comandos Git:

Si queremos eliminar un archivo de commit :

```
git rm --cached nombre_archivo
```

Si queremos utilizar Visual Studio Code como editor por defecto de Git:

```
git config --global core.editor "code --wait"
```

Si queremos modificar el último comentario realizado en un commit:

```
git commit --amend
```

Para deshacer el último commit :

```
git reset HEAD~1
```

Crear una rama:

```
git branch cambio-estilo
```

Para ver las ramas existentes:

```
git branch
```

Para cambiar el nombre de una rama (hay que estar en la rama a modificar):

```
git Branch -m nuevo-nombre
```

Para cambiar el nombre de una rama (desde cualquier rama):

```
git Branch -m nombre-actual nuevo-nombre
```

Para cambiar de rama :

```
git checkout nueva-rama
```

Para eliminar una rama :

```
git branch -d nombre-rama
```

Para visualizar logs de una rama:

```
git log rama
```

Para visualizar logs de una rama:

```
git log rama
```

Para visualizar logs solo la primera linea:

```
git log --oneline
```

Para visualizar logs con los cambios realizados:

```
git log -p
```

Fusionar ramas (hay que estar en la rama que va a recibir la fusión):

```
git merge rama_a_fusionar
```

Se pide:

1. Instalar Git en tu máquina Windows
2. Realiza los pasos indicados en este documento para trabajar con Git.
3. Realiza la actividad en clase de elaboración de recetas
4. Elabora un documento con las actividades realizadas