

# VERSIONAMIENTO DE CÓDIGO CON



# git



## Estructura de contenidos

|   | Pág. |
|---|------|
| Introducción .....  | 3    |
| Mapa de contenido .....   | 4    |
| 1. Generalidades .....  | 5    |
| 1.1 Historia de Git.....  | 5    |
| 1.2 Características principales de Git.....                         | 6    |
| 2. Conceptos básicos de control de cambios de software .....        | 6    |
| 3. Instalación de Git.....  | 8    |
| 4. Primeros pasos con Git.....                                      | 10   |
| 4.1 Creación del repositorio local .....                            | 10   |
| 4.2 Adición de archivos para controlar cambios .....                | 11   |
| 4.3. Aplicación de cambios y revisión del estado del proyecto ..... | 12   |
| 4.4. Aplicación y seguimiento a nuevos cambios .....                | 14   |
| 4.5. Navegación a través del historial de cambios .....             | 17   |
| 4.6. Uso de Git con repositorios remotos .....                      | 20   |
| 5. Uso de Git en Netbeans .....                                     | 21   |
| 5.1. Creación de una aplicación de pruebas .....                    | 22   |
| 5.2. Creación del repositorio.....                                  | 25   |
| 5.3. Primera confirmación o commit.....                             | 26   |
| 5.4. Aplicación de nuevos cambios al proyecto.....                  | 31   |
| 5.5. Revisión de los cambios aplicados.....                         | 33   |
| Glosario .....  | 38   |
| Bibliografía.....   | 39   |
| Control del documento .....   | 40   |

## Introducción

La construcción de aplicaciones empresariales requiere con frecuencia del trabajo de equipos de desarrolladores que trabajan simultáneamente. Lo anterior requiere una coordinación precisa de lo contrario puede suceder que cada desarrollador tenga una versión distinta del proyecto y se presenten problemas al momento de realizar el empalme.

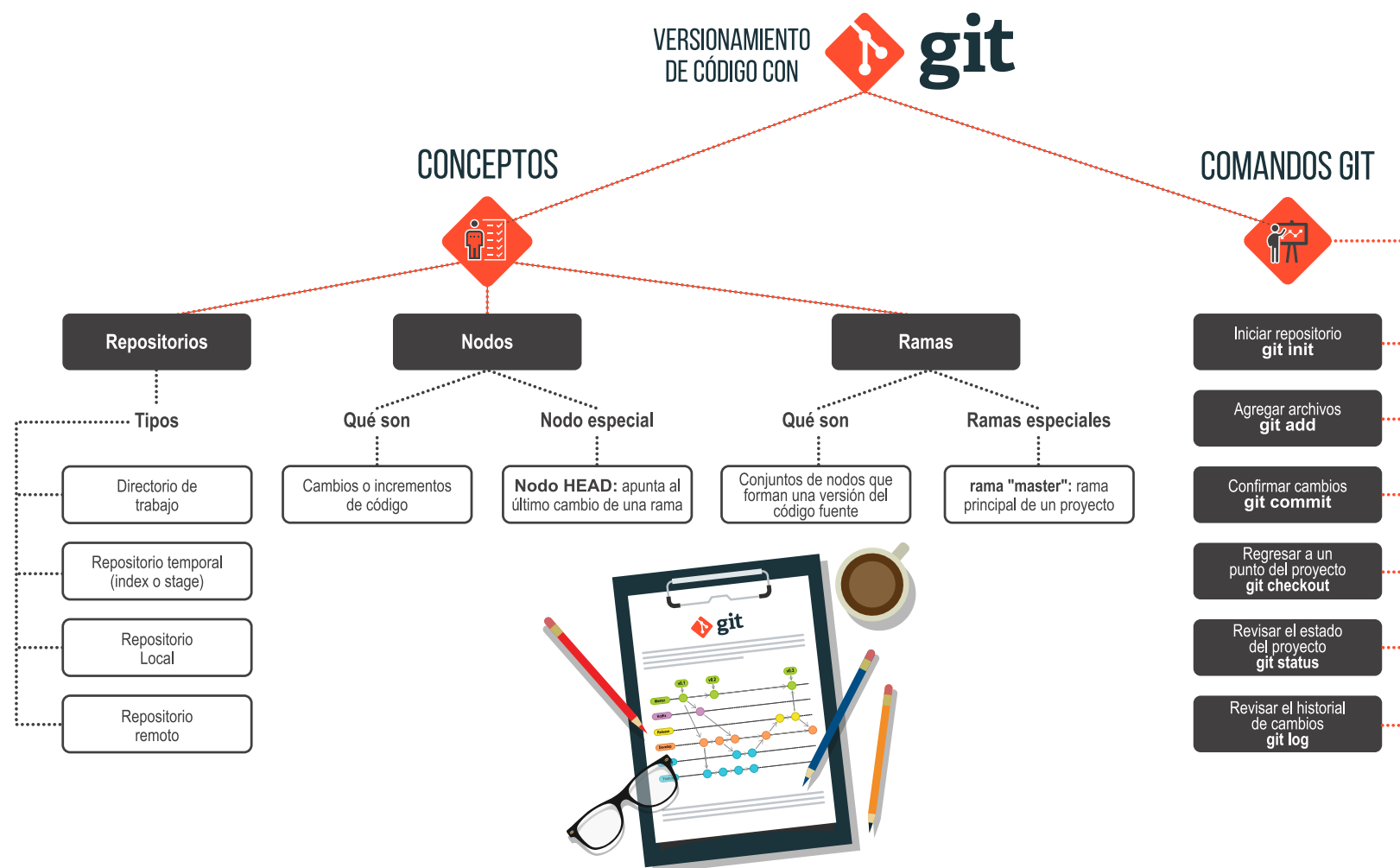
Por otra parte, el desarrollo de software es un arte que se lleva a cabo de manera gradual a través de incrementos que se dan sobre un código que ha tenido pruebas o realimentación.

Es así como surge **Versión Control System** (VCS) o sistemas de control de versiones que como su nombre lo indica llevan el control del código fuente que se va liberando de tal forma que los desarrolladores puedan conocer la evolución del desarrollo a través de las herramientas que suministran estos sistemas.

El presente recurso realiza un ejercicio práctico de la utilización del sistema de control de versiones llamado Git el cual se puede trabajar desde la consola y también se puede trabajar de manera integrada con el entorno de desarrollo Netbeans.



## Mapa de contenido



## Desarrollo de contenidos

### 1. Generalidades

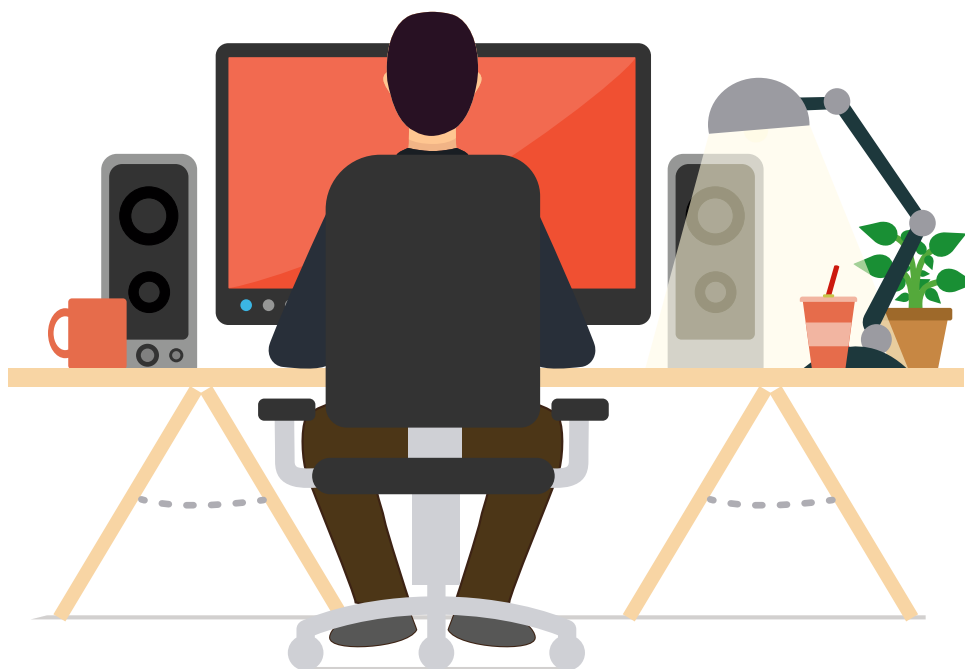
El desarrollo de software está constantemente enfrentado al cambio. Los negocios están en constante evolución y requieren que el sistema de información soporte o refleje los cambios que deben hacer las empresas u organizaciones en sus procesos para mantenerse competitivas o innovadoras.

Sin embargo, los sistemas de información requieren por un lado la coordinación entre los diferentes desarrolladores para que los cambios de uno no afecten negativamente el trabajo de otro miembro del equipo.

Por otro lado, se requiere que los cambios introducidos se puedan revertir en caso que estos últimos no hayan sido exitosos o ya no sean requeridos. También es importante para los desarrolladores revisar los cambios que se han introducido a lo largo de la vida del proyecto.

#### 1.1 Historia de Git

Git fue desarrollado por Linus Torvalds, el creador del sistema operativo Linux, en el año 2005 como respuesta a la falta de funcionalidad que tenía el software que usaba en ese momento llamado VCS para llevar control del desarrollo del kernel del sistema Linux.



## 1.2 Características principales de Git

Las ventajas de Git sobre otros sistemas de control de código son:

- a. Control de código distribuido: permite a equipos de desarrolladores trabajar de manera remota sobre un mismo proyecto.
- b. Permite trabajar fuera de línea: permite hacer contribuciones de código de manera local y luego replicarlos a un repositorio remoto.
- c. De fuente abierta y gratuito: no se requiere licenciamiento y su código fuente está disponible sin cargo.
- d. Está basado en las ideas de otros sistemas que tuvieron éxito como BitKeeper y Monotone.
- e. Viene integrado en varios de los más usados IDE's como Netbeans y Eclipse.



## 2. Conceptos básicos de control de cambios de software

Una forma de comprender el concepto que usa Git para manejar los cambios es ver el software como un árbol, en el sentido matemático del término, con un único nodo padre y uno o muchos nodos hijos como se muestra a continuación:

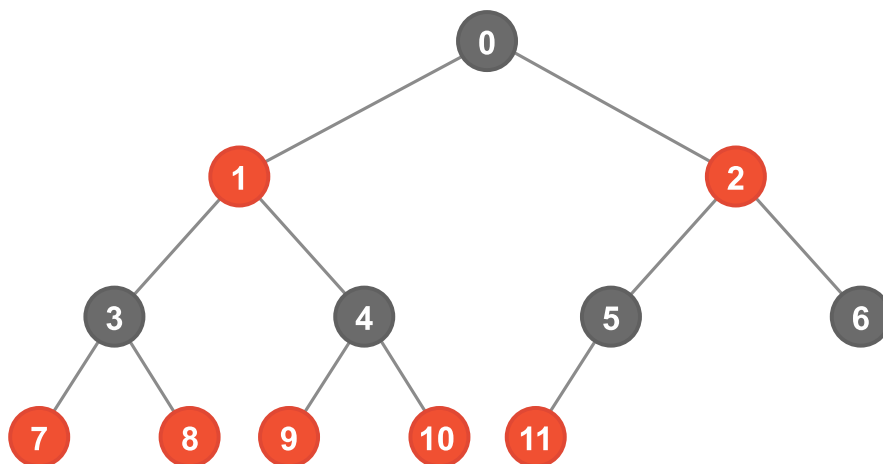


Figura 2.1 Estructura de árbol con único nodo padre.

Para Git el nodo cero es el nodo padre y es la versión inicial del software que se quiere controlar. A partir de ese estado inicial se pueden realizar incrementos que son registrados cuando se confirman a través de la opción que suministra Git para tal efecto.

Cada incremento, también llamado commit, tiene un identificador único y un nombre. Además, cada incremento o commit es un nuevo nodo del árbol.

Los conjuntos de nodos que incluyen el nodo padre y terminan en un nodo sin hijos se denominan ramas (branch).

Git permite dar nombre a las ramas. En términos de software las ramas son las versiones del código que se están desarrollando. La rama principal se llama “master” y Git la identifica una vez se genera el repositorio. Posteriormente se pueden crear ramas cuyos nombres son asignados por los usuarios. En la figura 2.1 se pueden observar las siguientes ramas: {0,1,3,7}, {0,1,3,8}, {0,1,4,9}, {0,1,4,10}, {0,2,5,11} y {0,2,6}.

Además del nodo padre existen otros nodos especiales y son aquellos que están al final de cada rama como los nodos 6,7,8,9,10 y 11 de la figura 2.1. Estos nodos toman el nombre de HEAD o cabeza de la rama. Estos nodos son importantes y de ahí que tengan un nombre especial porque indican el último cambio realizado a la rama.

A continuación, un ejemplo de un árbol típico de un proyecto Git:

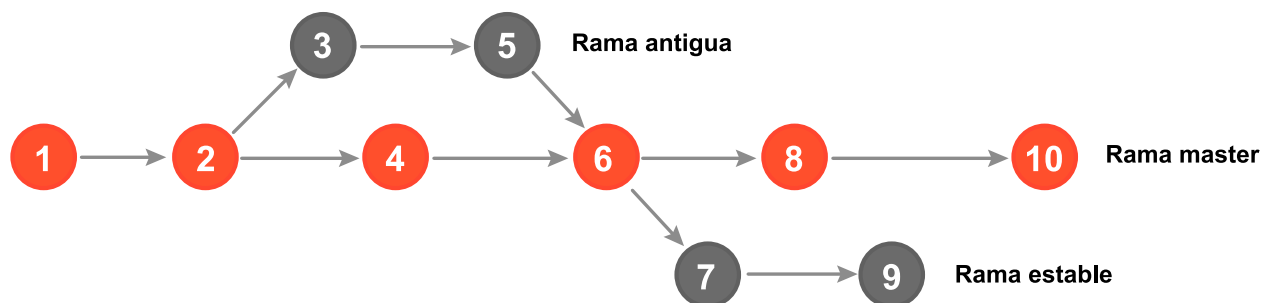


Figura 2.2. Un árbol típico con Git

De la figura 2.2. se pueden observar dos ramas actuales:

- La rama {1,2,4,6,8,10} que es la rama principal o master.
- La rama {1,2,4,6,7,9} que se llama la rama “estable”.

Se puede observar también que en algún momento de la historia del proyecto existió una rama {1,2,3,5} que posteriormente se unió a la rama master.

Hasta este punto se describieron los conceptos sobre los cuales está diseñado Git. En adelante se hará la instalación del software y se harán ejercicios prácticos con la herramienta.

### 3. Instalación de Git

Para instalar Git se descarga del sitio oficial <https://git-scm.com/downloads> como se muestra a continuación:

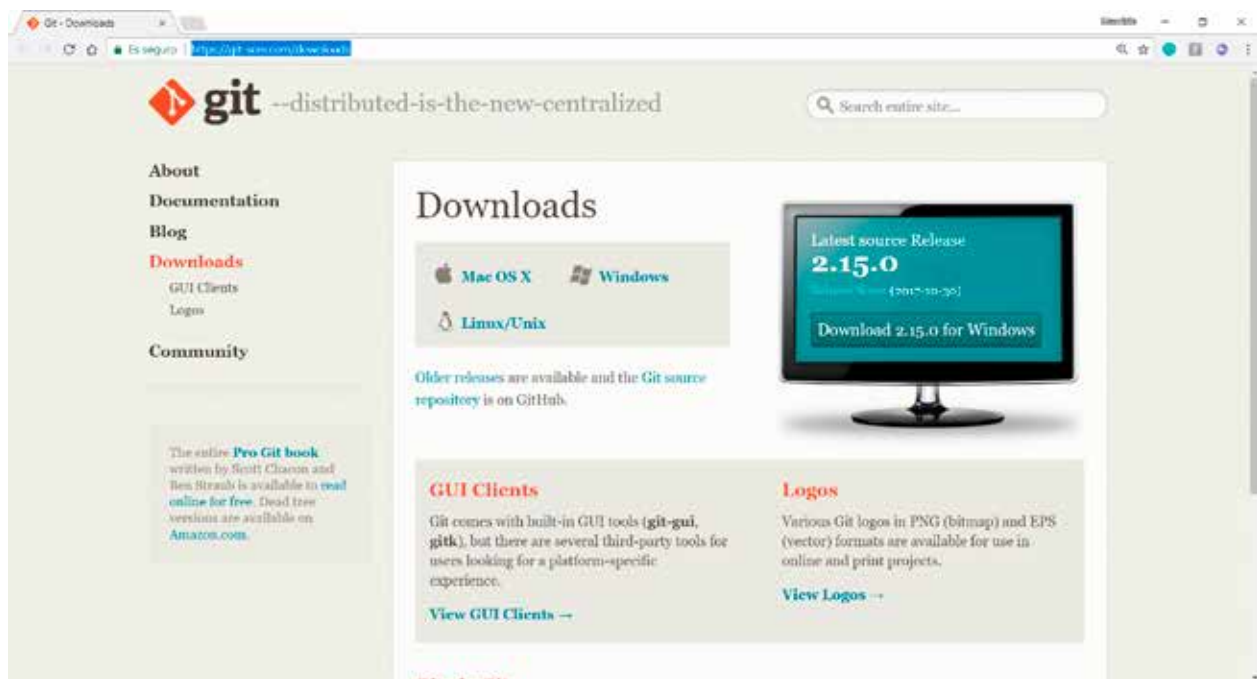


Figura 3.1. Página oficial de Git.

Se debe elegir el tipo de sistema operativo y por último se descarga y se instala el software haciendo clic sobre el instalador. En el proceso de instalación se deben dejar las opciones por defecto. A continuación, se hacen algunas observaciones.

Se debe permitir usar Git desde la línea de comando. Al momento de la instalación la siguiente página debe quedar así:



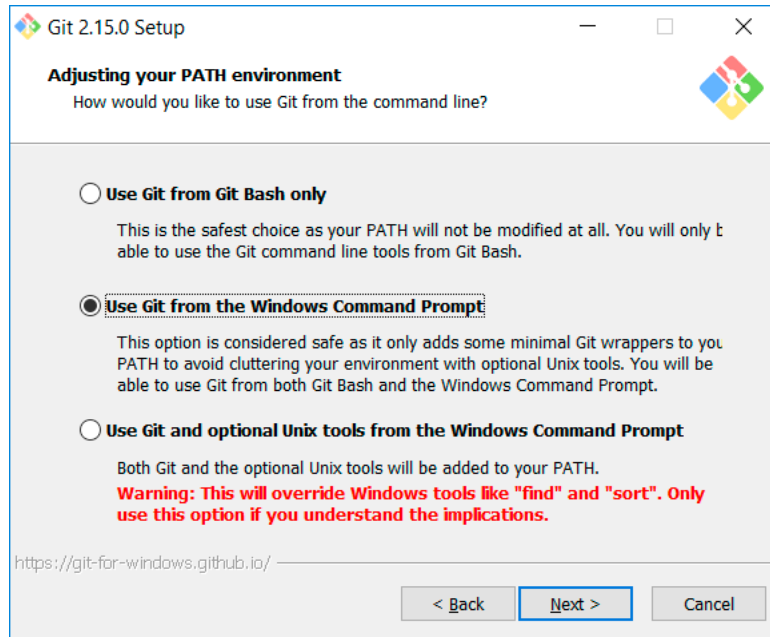


Figura 3.2. Ajustes al PATH al momento de instalar Git.

Lo anterior para poder ejecutar Git desde la línea de comando del sistema operativo.

También se debe seleccionar la herramienta SSH incluida en Git como aparece a continuación:

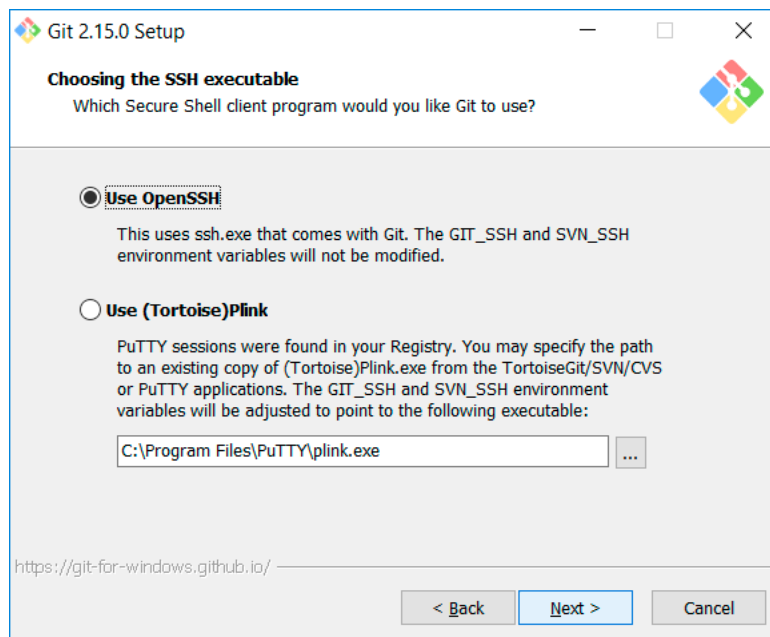
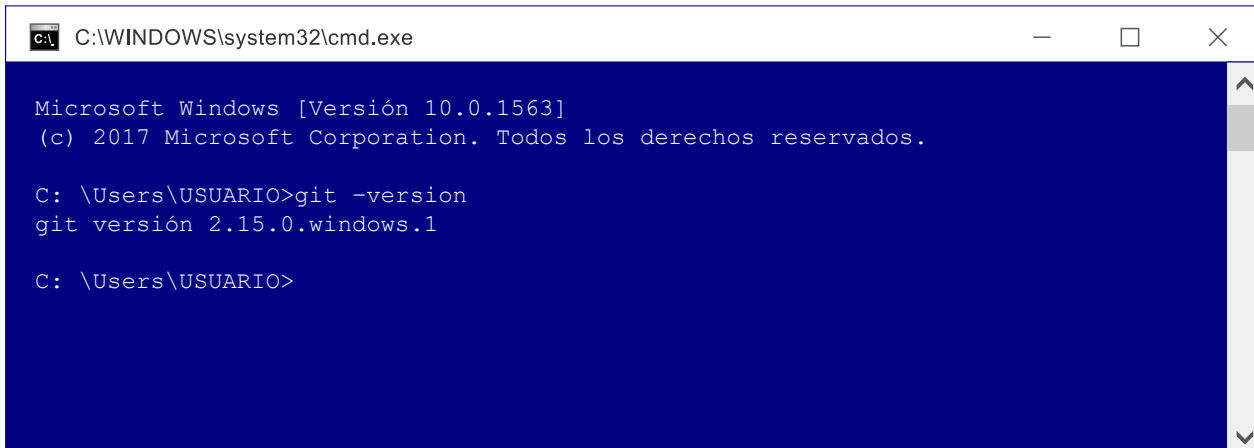


Figura 3.3. Selección de SSH.

Para revisar que Git esté instalado se abre una ventana de línea de comandos y se introduce la orden: `git - version` así:



```
C:\WINDOWS\system32\cmd.exe

Microsoft Windows [Versión 10.0.1563]
(c) 2017 Microsoft Corporation. Todos los derechos reservados.

C: \Users\USUARIO>git -version
git versión 2.15.0.windows.1

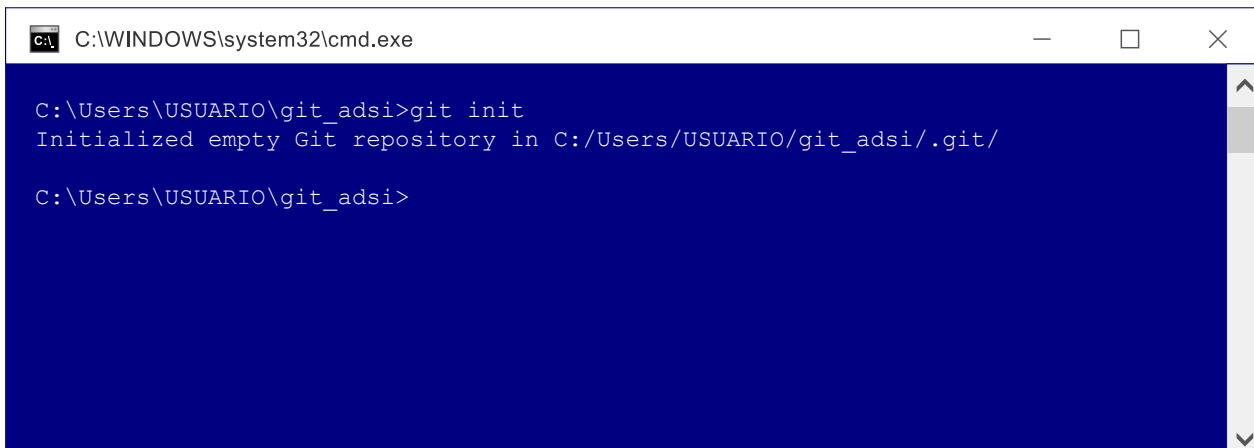
C: \Users\USUARIO>
```

Figura 3.4. Revisión de la instalación de Git.

## 4. Primeros pasos con Git

### 4.1 Creación del repositorio local

El primer paso consiste en determinar cuál va a ser el código al cual se le realizará un seguimiento o control. Este código debe estar en una carpeta del sistema operativo. Para esto se creará la carpeta llamada “git\_adsi”. Luego de ubicarse sobre este directorio se usa el comando “git init” así:



```
C:\WINDOWS\system32\cmd.exe

C:\Users\USUARIO\git_adsi>git init
Initialized empty Git repository in C:/Users/USUARIO/git_adsi/.git/

C:\Users\USUARIO\git_adsi>
```

Figura 4.1. Inicialización de un repositorio local en Git.

En este punto Git generó un directorio llamado “.git” donde ubicará los cambios que se realicen.

## 4.2 Adición de archivos para controlar cambios

En el numeral 4.1. se creó el repositorio o directorio donde Git lleva registro de los cambios. El siguiente paso consiste en definir cuáles archivos se van a controlar. Para lo anterior se usa el comando “git add <nombre\_de\_archivo>”. Se puede agregar archivo de manera individual: “git add <archivo>” y también se puede agregar todo un directorio con el comando “git add”.

Antes de agregar los archivos, se debe crear uno para realizar las pruebas. Para este ejercicio se usará un archivo de texto llamado “adsi.txt” cuyo contenido es una sola línea de texto así:

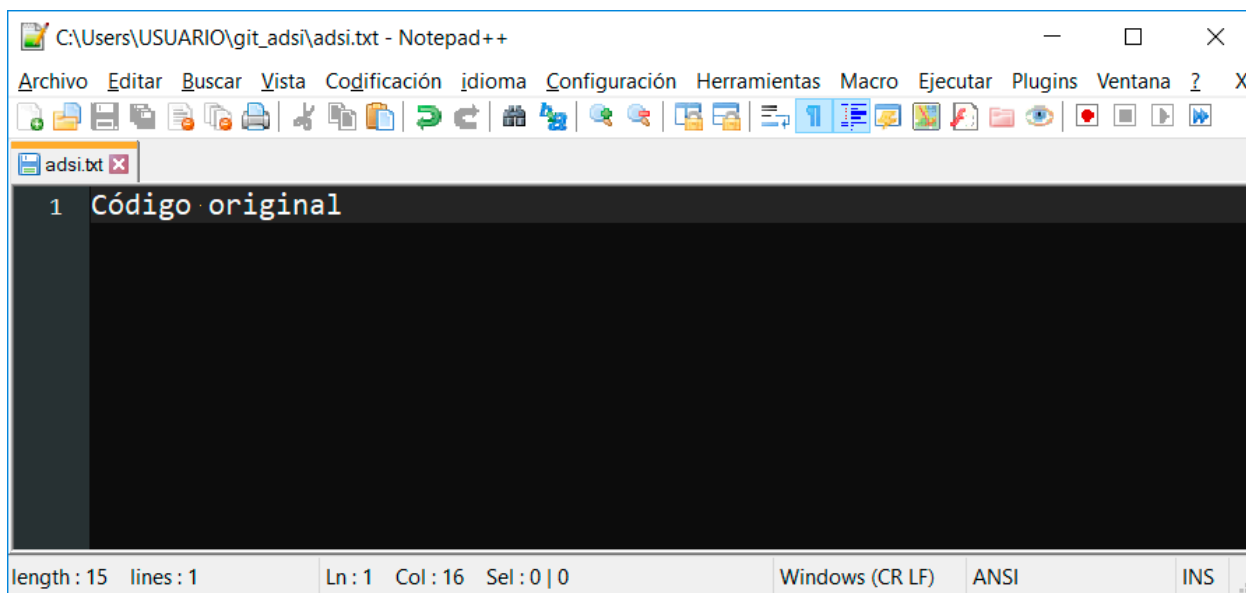


Figura 4.2. Archivo original al cual se le controlarán los cambios con Git.

En este punto el directorio “git\_adsi” contiene lo siguiente:

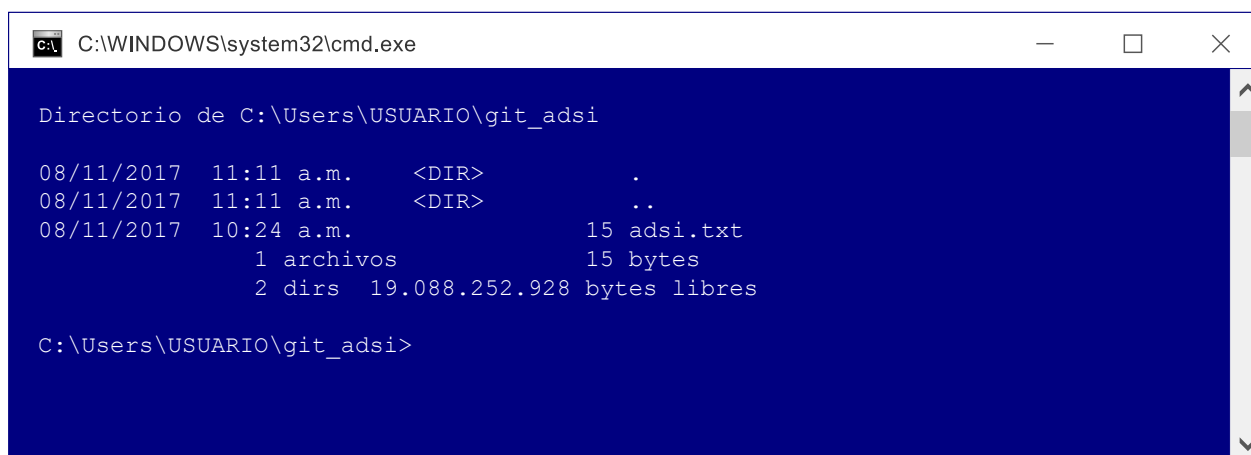
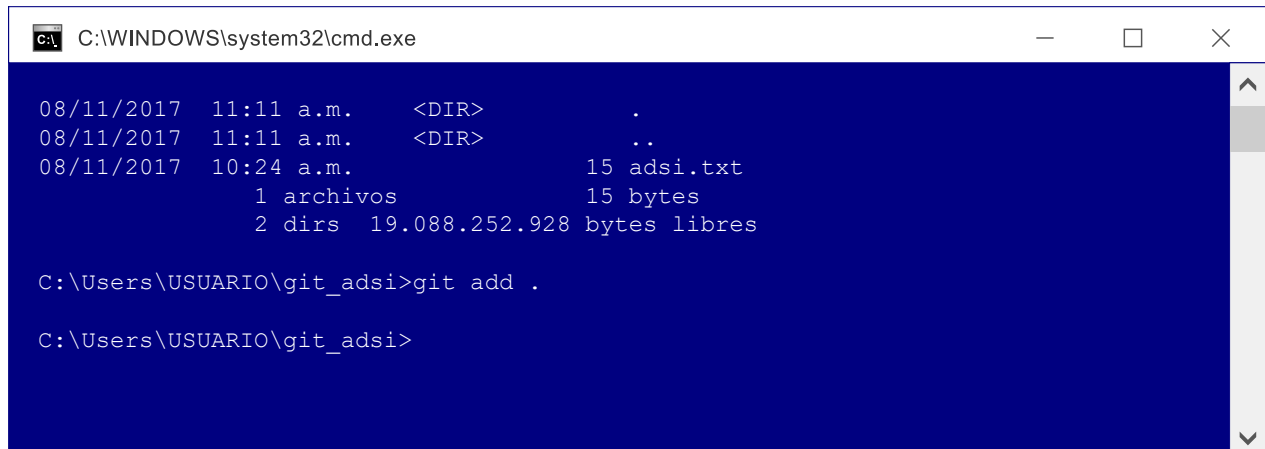


Figura 4.3. Contenido del directorio.

Para marcar este archivo y todos los que estén en el directorio actual se procede así:



```
C:\WINDOWS\system32\cmd.exe

08/11/2017  11:11 a.m.    <DIR>      .
08/11/2017  11:11 a.m.    <DIR>      ..
08/11/2017  10:24 a.m.                15 adsi.txt
                1 archivos                15 bytes
                2 dirs 19.088.252.928 bytes libres

C:\Users\USUARIO\git_adsi>git add .

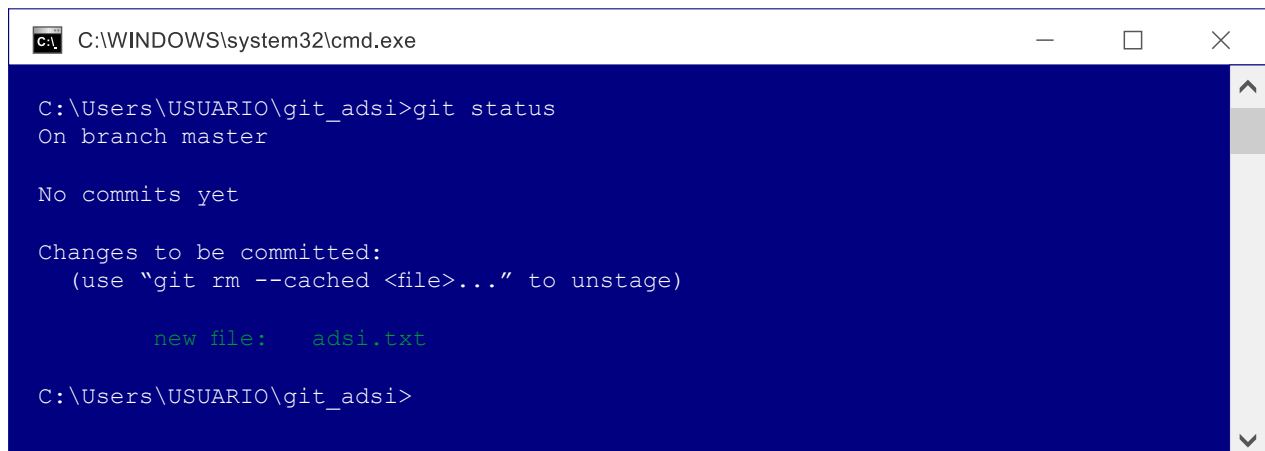
C:\Users\USUARIO\git_adsi>
```

Figura 4.4. Inclusión inicial de archivos.

### 4.3. Aplicación de cambios y revisión del estado del proyecto

En el numeral 4.2 se usó el comando “git add” para agregar archivos. Sin embargo, Git no aplica automáticamente los cambios, sino que los almacena temporalmente en el repositorio o árbol “index”

Se pueden revisar el estado de los cambios pendientes con el comando “git status” así:



```
C:\WINDOWS\system32\cmd.exe

C:\Users\USUARIO\git_adsi>git status
On branch master

No commits yet

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

        new file:   adsi.txt

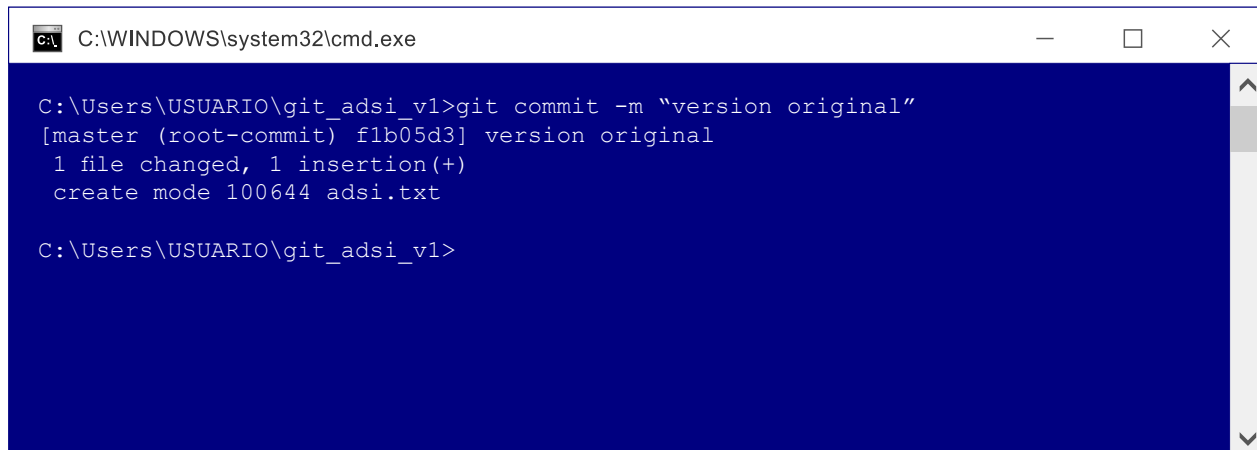
C:\Users\USUARIO\git_adsi>
```

Figura 4.5. Revisión de cambios con el comando “git status”.

Se puede observar que la consola muestra el archivo “adsi.txt” como uno nuevo e indica que forma parte de los cambios pendientes.

Para confirmar los cambios se usa el comando “git commit”. Este comando lo que hace es pasar los cambios de un repositorio “index” al repositorio definitivo o “head”.

El comando “git commit” acepta comentarios usando la opción “-m” así: “git commit -m <mensaje> “. Para aplicar los cambios al ejercicio en curso se procede así:



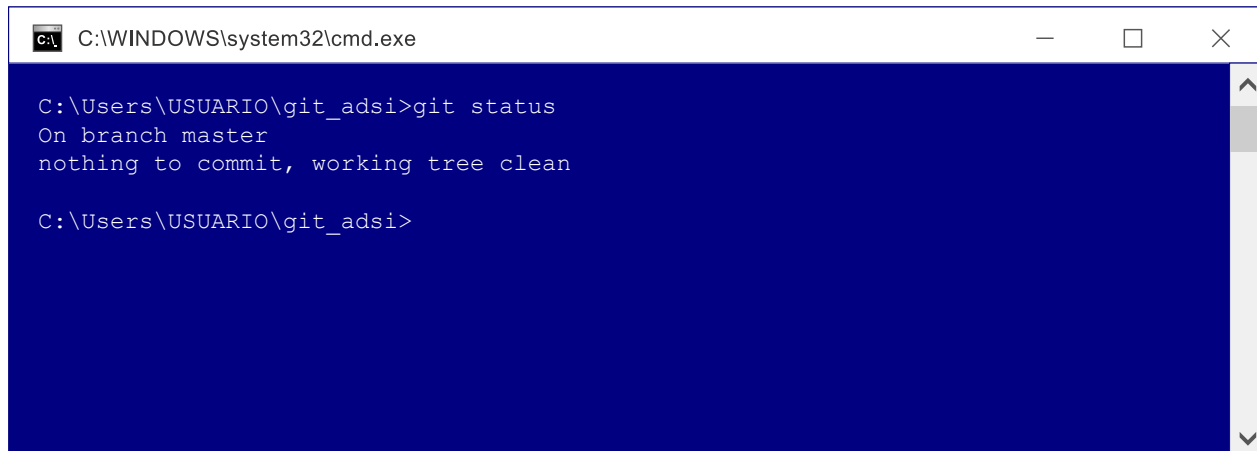
```
C:\WINDOWS\system32\cmd.exe

C:\Users\USUARIO\git_adsi_v1>git commit -m "version original"
[master (root-commit) flb05d3] version original
1 file changed, 1 insertion(+)
create mode 100644 adsi.txt

C:\Users\USUARIO\git_adsi_v1>
```

Figura 4.6. Aplicación de los cambios realizados.

Se puede en este momento volver a revisar el estado o “status” así:



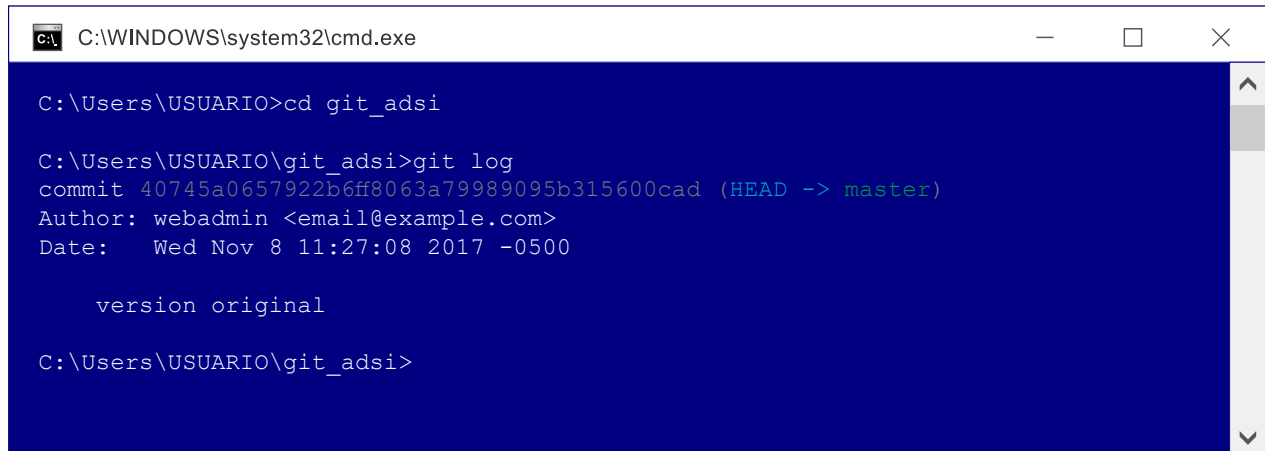
```
C:\WINDOWS\system32\cmd.exe

C:\Users\USUARIO\git_adsi>git status
On branch master
nothing to commit, working tree clean

C:\Users\USUARIO\git_adsi>
```

Figura 4.7. Aplicación de los cambios realizados.

Se puede observar que no hay cambios para aplicar. En este punto se puede revisar el historial de cambios con el comando “git log” así:



```

C:\WINDOWS\system32\cmd.exe

C:\Users\USUARIO>cd git_adsi

C:\Users\USUARIO\git_adsi>git log
commit 40745a0657922b6ff8063a79989095b315600cad (HEAD -> master)
Author: webadmin <email@example.com>
Date:   Wed Nov 8 11:27:08 2017 -0500

    version original

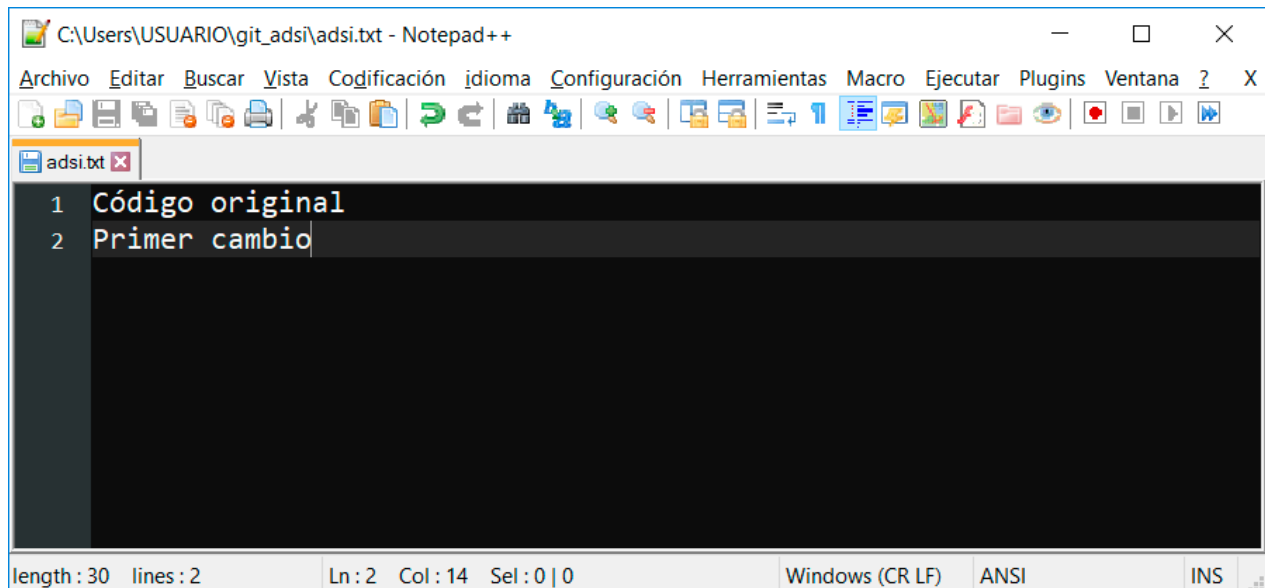
C:\Users\USUARIO\git_adsi>
  
```

Figura 4.8. Revisión del historial de cambios.

#### 4.4. Aplicación y seguimiento a nuevos cambios

Del numeral 4.2. al 4.3. se creó el repositorio y se hizo un primer “commit” con los archivos originales del proyecto. Ahora se van a realizar modificaciones al archivo inicial “adsi.txt” y mostrar cómo Git lleva el control de los mismos.

Para lo anterior se edita el archivo y se le agrega una nueva línea como se muestra a continuación:



```

C:\Users\USUARIO\git_adsi\adsi.txt - Notepad++

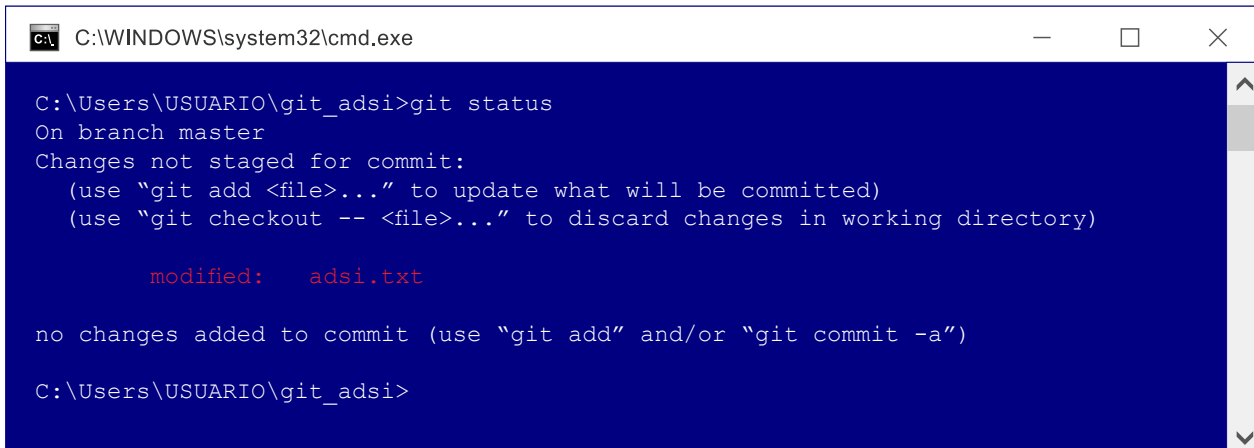
Archivo  Editar  Buscar  Vista  Codificación  Idioma  Configuración  Herramientas  Macro  Ejecutar  Plugins  Ventana  ?  X

adsi.txt x
1 Código original
2 Primer cambio

length: 30  lines: 2  Ln: 2  Col: 14  Sel: 0 | 0  Windows (CR LF)  ANSI  INS
  
```

Figura 4.9. Introducción del primer cambio al archivo original.

Una vez se guarden los cambios se puede revisar el estado del proyecto con el comando “git status” así:



```
C:\WINDOWS\system32\cmd.exe

C:\Users\USUARIO\git_adsi>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   adsi.txt

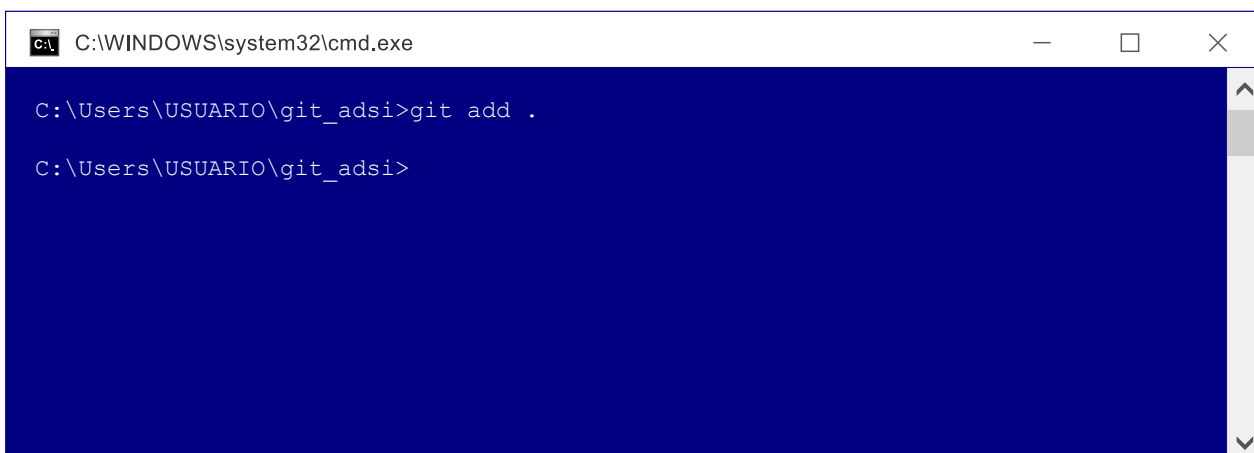
no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\USUARIO\git_adsi>
```

Figura 4.10. Revisión del estado del proyecto con “git status”.

Se puede observar que Git detectó el cambio y lo muestra en la sección de cambios por aplicar. En este punto el sistema permite o descarta esos cambios con el comando “git checkout” o agrega los cambios a la cola de trabajo para posteriormente confirmarlos.

En este ejemplo se procederá primero a incluir los cambios en la cola de trabajo con el comando “git add” así:

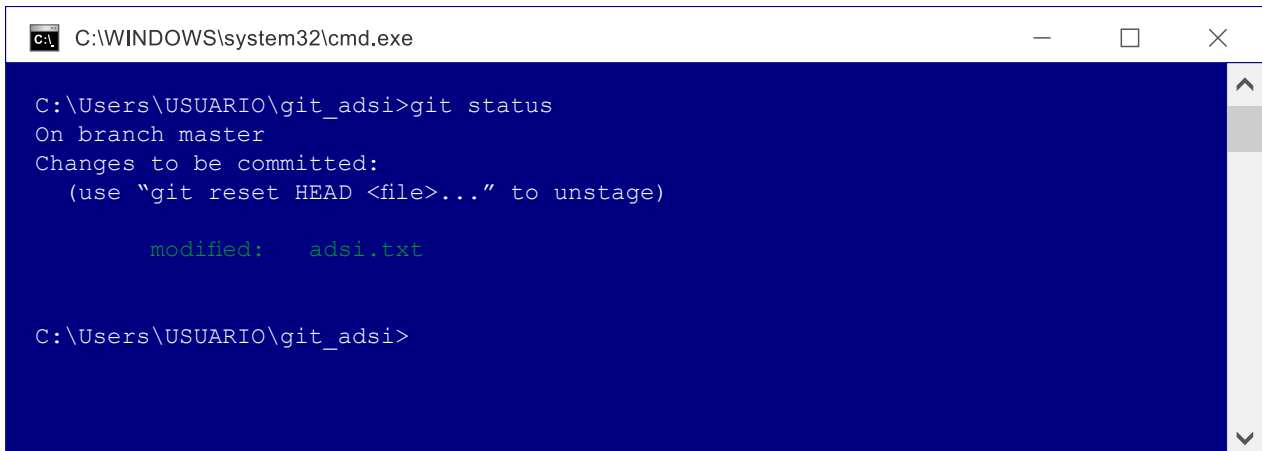


```
C:\WINDOWS\system32\cmd.exe

C:\Users\USUARIO\git_adsi>git add .
C:\Users\USUARIO\git_adsi>
```

Figura 4.11. Inclusión de nuevos cambios en la cola de trabajo.

Al revisar el estado del proyecto con el comando “git status” aparece lo siguiente:



```
C:\WINDOWS\system32\cmd.exe

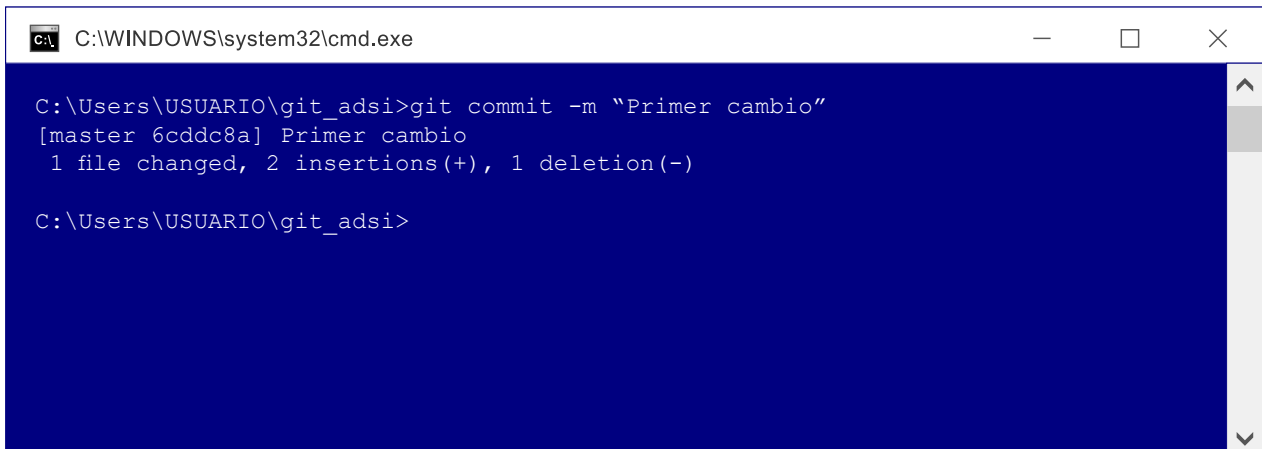
C:\Users\USUARIO\git_adsi>git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   adsi.txt

C:\Users\USUARIO\git_adsi>
```

Figura 4.12. Estado del proyecto.

Luego se confirman los cambios con el comando “git commit” y le agrega un comentario así:



```
C:\WINDOWS\system32\cmd.exe

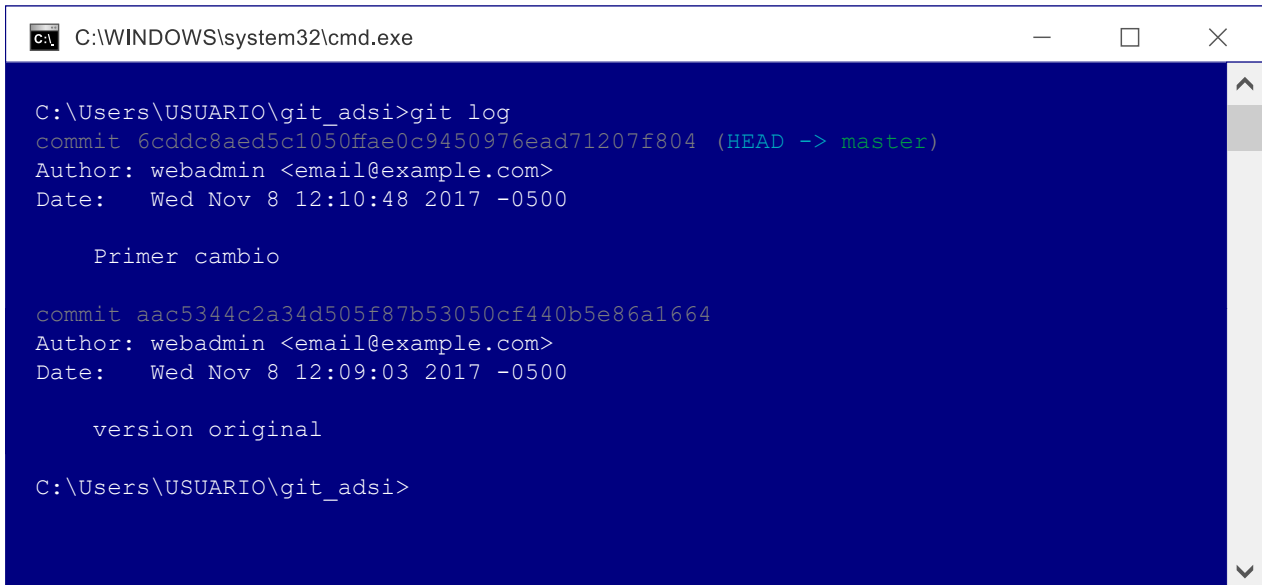
C:\Users\USUARIO\git_adsi>git commit -m "Primer cambio"
[master 6cddc8a] Primer cambio
 1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\USUARIO\git_adsi>
```

Figura 4.13. Confirmación de los cambios.



Por último, se revisa el historial de cambios con el comando “git log” así:



```

C:\Users\USUARIO\git_adsi>git log
commit 6cddc8aed5c1050ffae0c9450976ead71207f804 (HEAD -> master)
Author: webadmin <email@example.com>
Date:   Wed Nov 8 12:10:48 2017 -0500

    Primer cambio

commit aac5344c2a34d505f87b53050cf440b5e86a1664
Author: webadmin <email@example.com>
Date:   Wed Nov 8 12:09:03 2017 -0500

    version original

C:\Users\USUARIO\git_adsi>

```

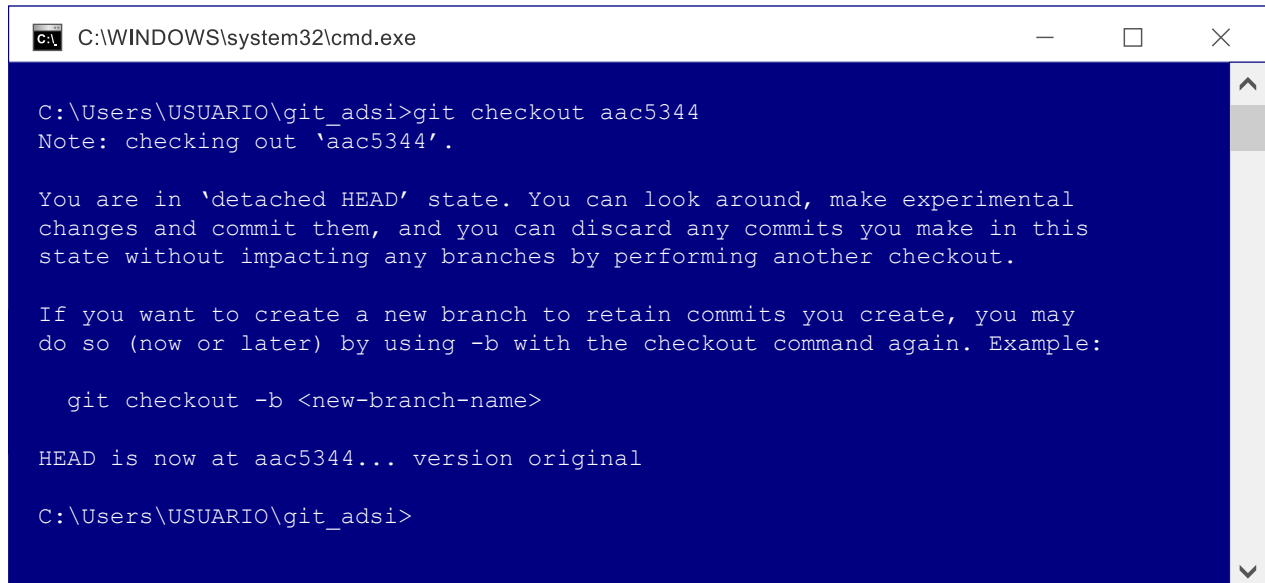
Figura 4.14. Historial de cambios del proyecto

Se puede observar que existen dos cambios registrados en el historial. El primero con el ID “6cddc8aed5c1050ffae0c9450976ead71207f804” y el segundo con el ID “aac5344c2a34d505f87b53050cf440b5e86a1664”. Los números de identificación son hash’s generados con el algoritmo SHA-1.

#### 4.5. Navegación a través del historial de cambios

Una de las características más importantes de Git es que permite regresar a un estado previo del proyecto, algo similar a un “viaje al pasado” que permite analizar los cambios o definitivamente regresar el proyecto ese punto de restauración.

El ejercicio del numeral 4.5. se generaron dos commit’s cada uno identificado con hasta 40 bytes. Para regresar o revisar un estado previo del proyecto se usa el comando “git checkout id-commit”. Si se requiere revisar el proyecto al momento que se dio el commit inicial se procede así:



```

C:\WINDOWS\system32\cmd.exe

C:\Users\USUARIO\git_adsi>git checkout aac5344
Note: checking out 'aac5344'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

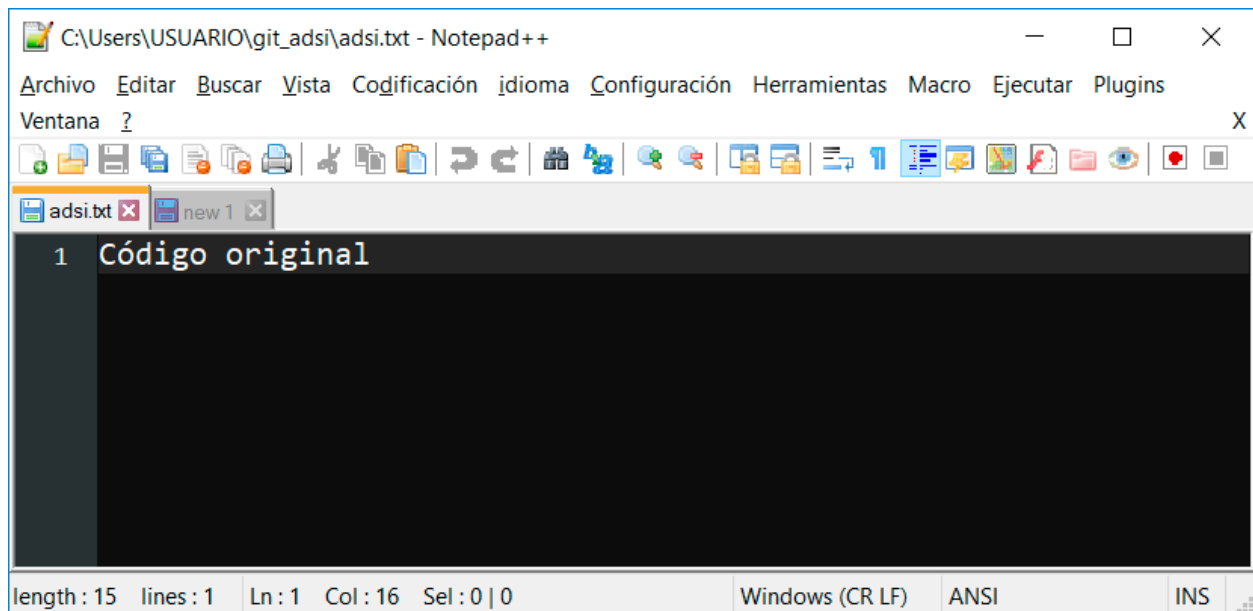
    git checkout -b <new-branch-name>

HEAD is now at aac5344... version original

C:\Users\USUARIO\git_adsi>
  
```

Figura 4.15. Uso de Git para regresar a un punto del proyecto.

Se puede observar que se pueden usar solo los primeros siete bytes del identificador del commit en lugar de todos los 40 bytes. En este punto si se edita el archivo “adsi.txt” aparece lo siguiente en pantalla:



```

C:\Users\USUARIO\git_adsi\adsi.txt - Notepad++
Archivo  Editar  Buscar  Vista  Codificación  Idioma  Configuración  Herramientas  Macro  Ejecutar  Plugins
Ventana  ?
adsi.txt  new 1

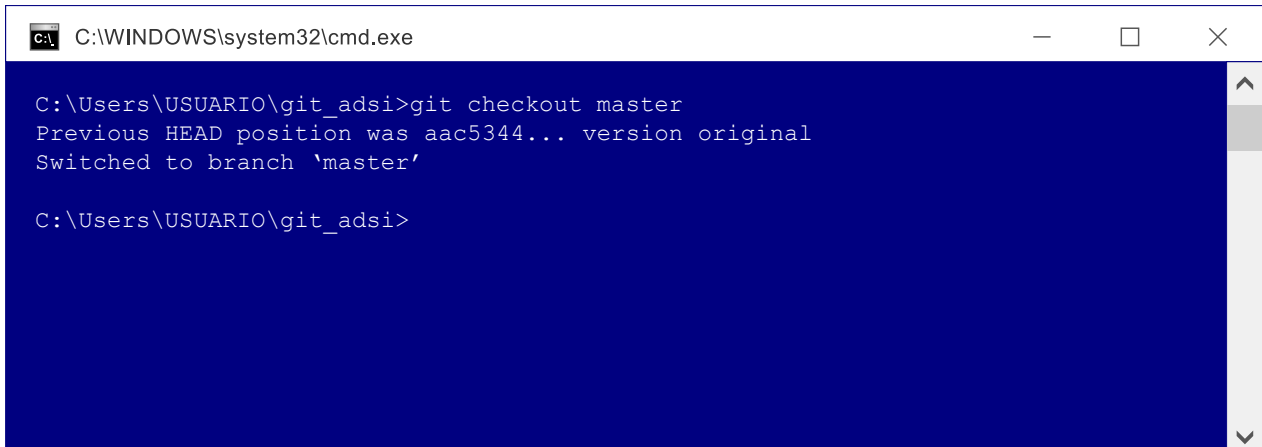
1  Código original

length: 15  lines: 1  Ln: 1  Col: 16  Sel: 0 | 0  Windows (CR LF)  ANSI  INS
  
```

Figura 4.16. Contenido del archivo adsi.txt en el commit original.

Se puede observar que el contenido del archivo es el original sin ningún cambio aplicado.

En este punto se puede estudiar el contenido del proyecto hasta el primer commit sin afectar el estado actual del mismo. Para retornar al último estado del proyecto se usa el comando “git checkout master” así:



```

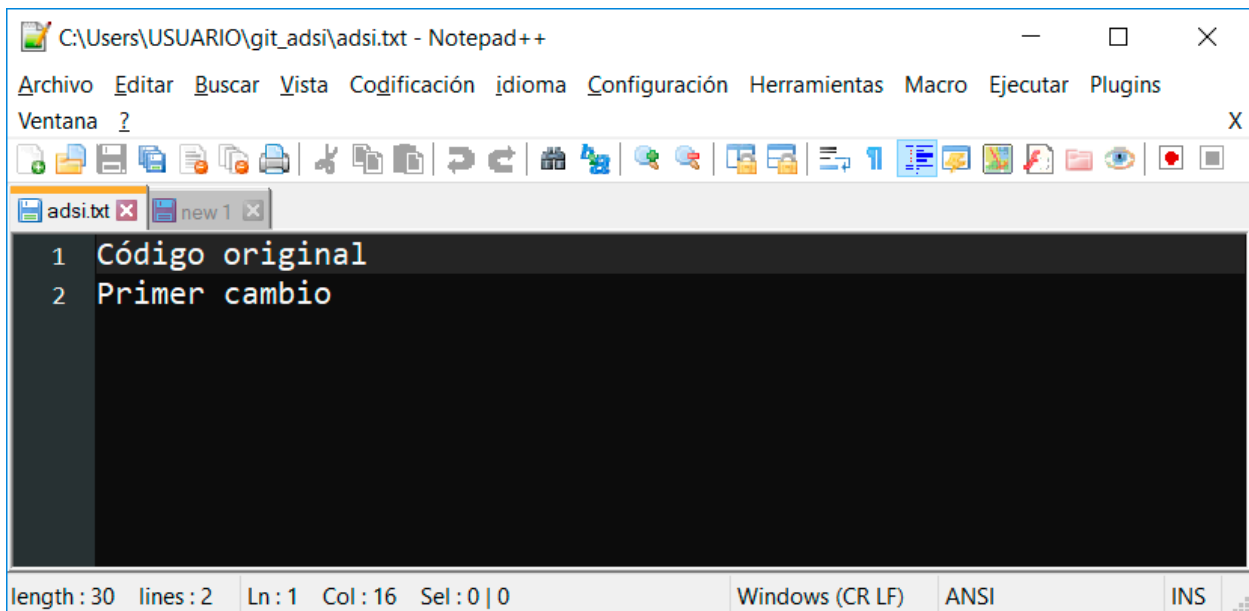
C:\WINDOWS\system32\cmd.exe

C:\Users\USUARIO\git_adi>git checkout master
Previous HEAD position was aac5344... version original
Switched to branch 'master'

C:\Users\USUARIO\git_adi>
  
```

Figura 4.17. Uso de Git para regresar al estado actual del proyecto.

En este punto el proyecto está ubicado en su última versión. Al revisar nuevamente el contenido del archivo “adi.txt” se muestra lo siguiente:



```

C:\Users\USUARIO\git_adi\adi.txt - Notepad++

Archivo  Editar  Buscar  Vista  Codificación  Idioma  Configuración  Herramientas  Macro  Ejecutar  Plugins
Ventana  ?

adi.txt x  new 1 x

1 Código original
2 Primer cambio

length: 30  lines: 2  Ln: 1  Col: 16  Sel: 0 | 0  Windows (CR LF)  ANSI  INS
  
```

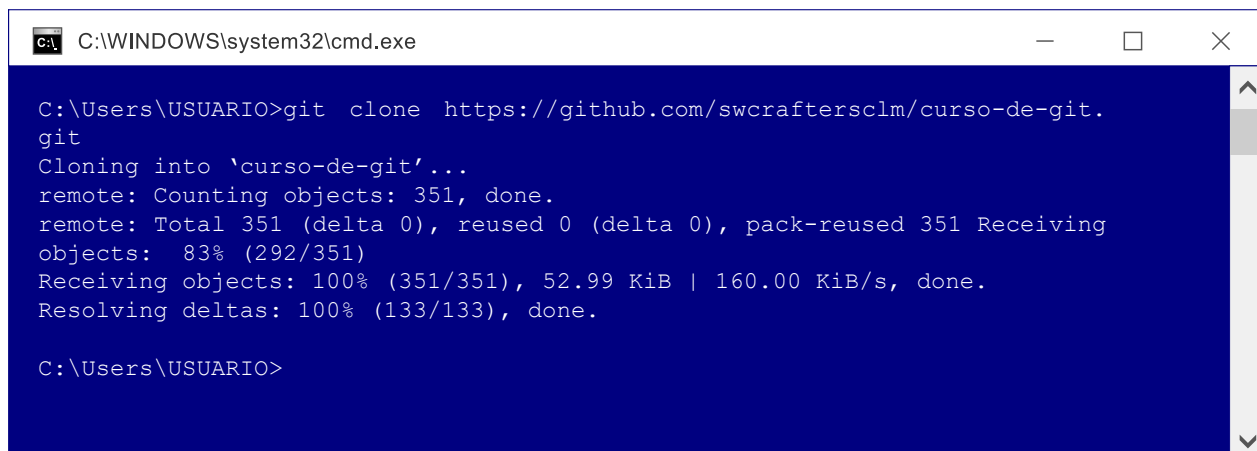
Figura 4.18. Archivo con los últimos cambios aplicados.

Se puede observar que el archivo de prueba tiene los cambios aplicados.

## 4.6. Uso de Git con repositorios remotos

En el numeral 4.1. se mostró la forma de crear un repositorio local. Este esquema se usa cuando se inicia desde cero un proyecto. Sin embargo, Git también permite trabajar con un proyecto ya existente que esté almacenado localmente o que esté en un sitio de la Internet.

Para copiar o iniciar un repositorio remoto en el directorio actual se usa el comando “git clone”.



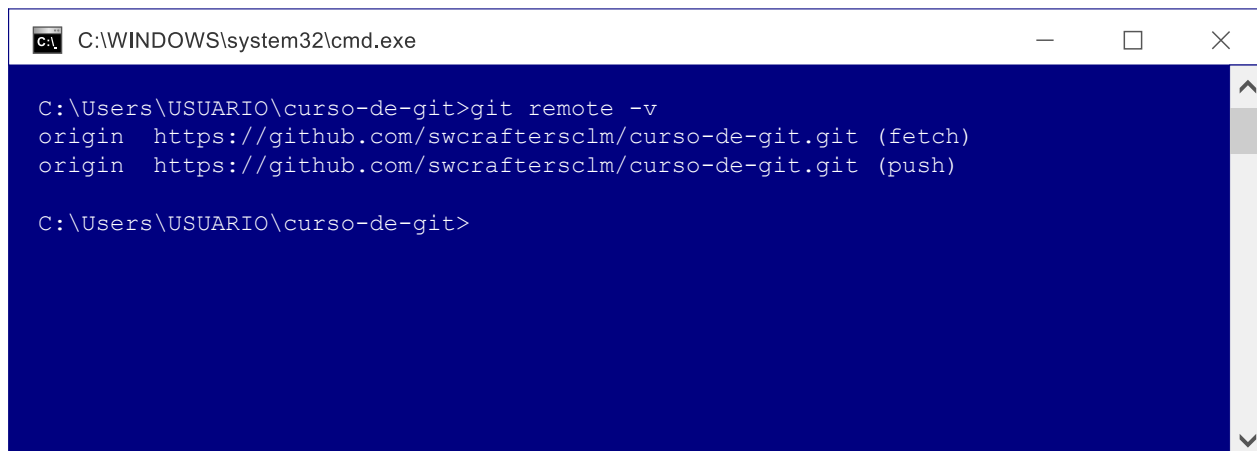
```
C:\WINDOWS\system32\cmd.exe

C:\Users\USUARIO>git clone https://github.com/swcraftersclm/curso-de-git.git
Cloning into 'curso-de-git'...
remote: Counting objects: 351, done.
remote: Total 351 (delta 0), reused 0 (delta 0), pack-reused 351
Receiving objects: 83% (292/351)
Receiving objects: 100% (351/351), 52.99 KiB | 160.00 KiB/s, done.
Resolving deltas: 100% (133/133), done.

C:\Users\USUARIO>
```

Figura 4.19. Ejemplo de copiado de un repositorio remoto.

En este ejemplo se usó un repositorio público alojado en el sitio [www.github.com](https://www.github.com). Una vez se cambia el directorio del proyecto se puede revisar el nombre del repositorio remoto con el comando “git remote -v” así:



```
C:\WINDOWS\system32\cmd.exe

C:\Users\USUARIO\curso-de-git>git remote -v
origin https://github.com/swcraftersclm/curso-de-git.git (fetch)
origin https://github.com/swcraftersclm/curso-de-git.git (push)

C:\Users\USUARIO\curso-de-git>
```

Figura 4.20. Revisión de los repositorios remotos.

## 5. Uso de Git en Netbeans

El entorno de desarrollo integrado o IDE Netbeans versión 8.2 tiene incluida una versión de Git. Para verificar que Git esté instalado se ejecuta Netbeans y se revisa la opción “Tools” -> “Options” → “Team”:

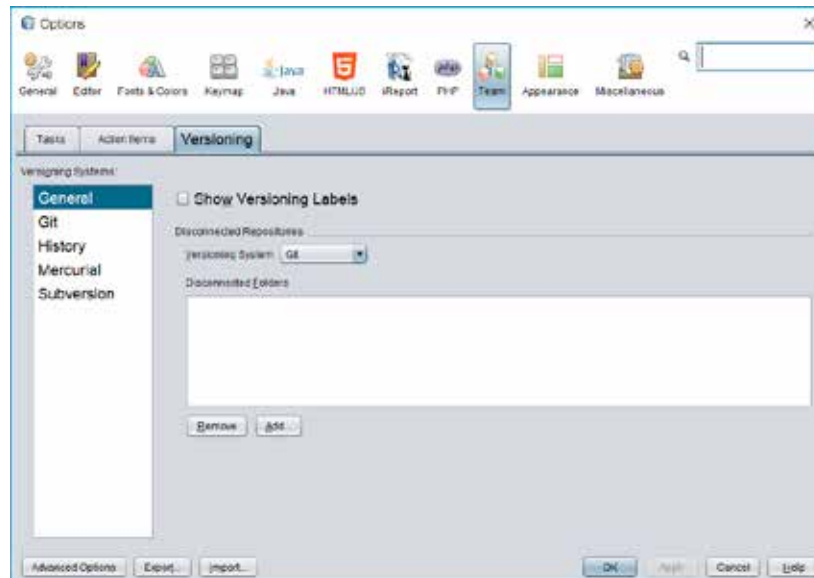


Figura 5.1. Verificación de la instalación de Git dentro de Netbeans.

Se puede observar que en la pestaña “Versioning” aparece Git dentro de los sistemas de versionamiento. Al dar clic sobre la opción Git del menú lateral aparece lo siguiente:

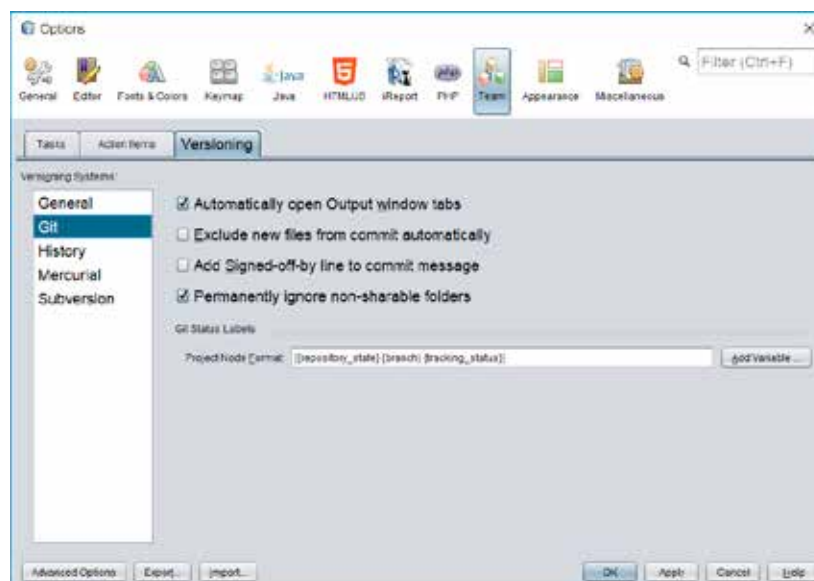


Figura 5.2. Verificación de la instalación de Git dentro de Netbeans.

## 5.1. Creación de una aplicación de pruebas

Para conocer la herramienta Git incluida en Netbeans se requiere crear una aplicación de pruebas. Para este ejercicio se creará una aplicación de tipo HTML5 con la ayuda de la mencionada herramienta. La aplicación se crea usando la opción “File” → “New project”. Una vez seleccionada aparece lo siguiente en pantalla:

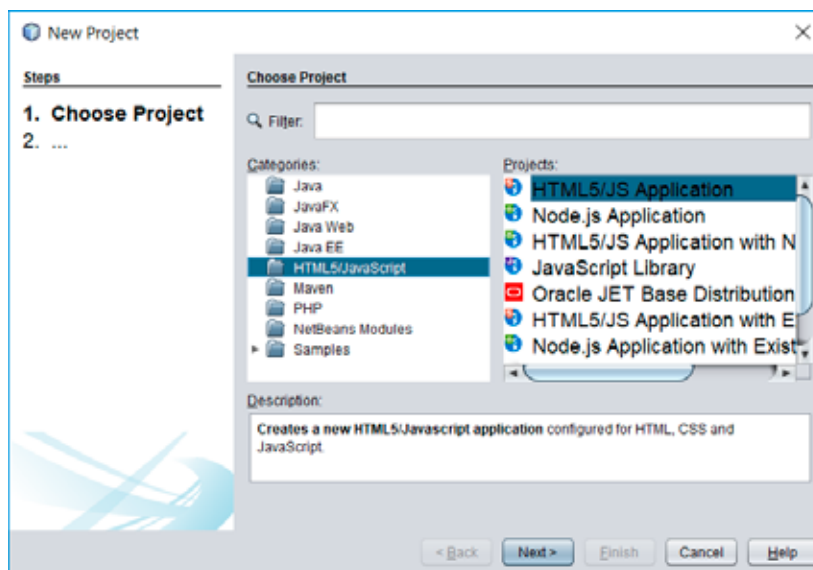


Figura 5.3. Creación de un proyecto HTML5 en Netbeans.

Se debe oprimir el botón “Next” y aparece lo siguiente en pantalla:

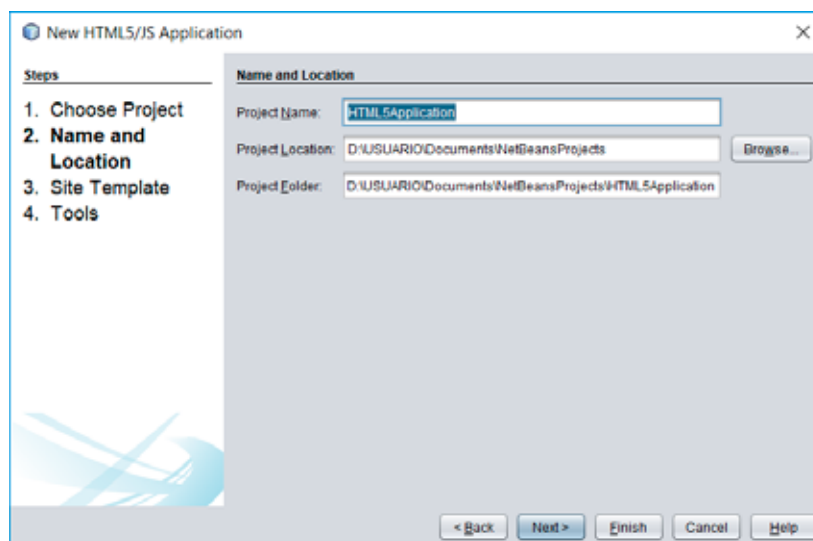


Figura 5.4. Creación de un proyecto HTML5 en Netbeans parte 2.

Se debe especificar un nombre de proyecto en el campo “Project Name”. Para este ejercicio el nombre será: “adsi\_git”. Luego se oprime el botón “Next”. Aparece lo siguiente en pantalla:

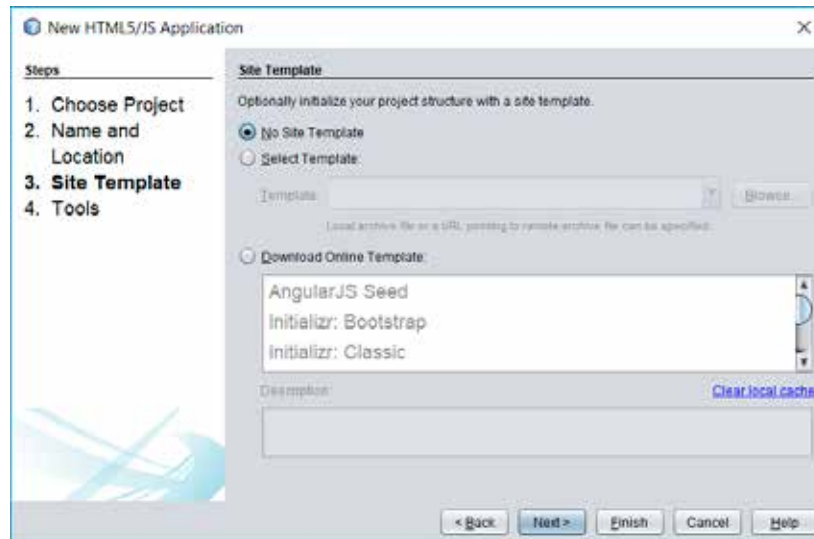


Figura 5.4. Creación de un proyecto HTML5 en Netbeans parte 3.

En esta pantalla no se selecciona ninguna opción y se oprime el botón “Next”. Aparece lo siguiente en pantalla:

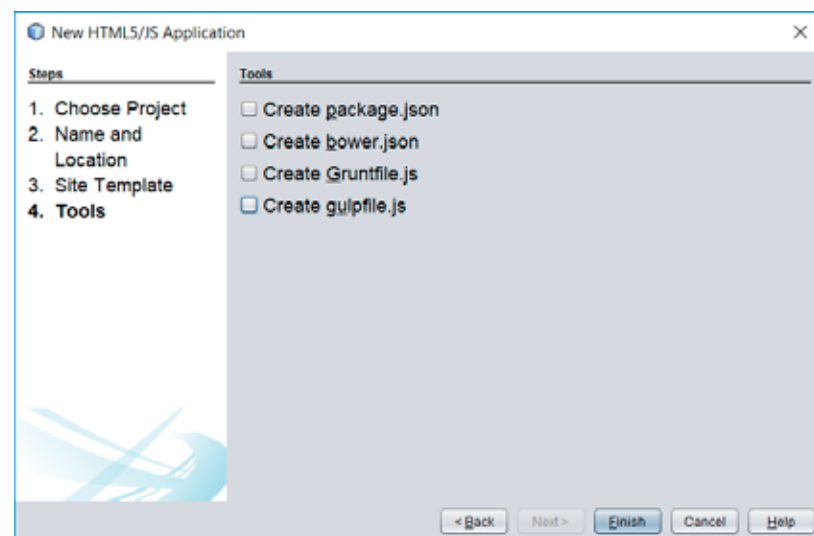


Figura 5.5. Creación de un proyecto HTML5 en Netbeans parte 4.

En esta pantalla se quitan las opciones seleccionadas por defecto. Por último se oprime el botón “Finish”. Aparece lo siguiente en pantalla:

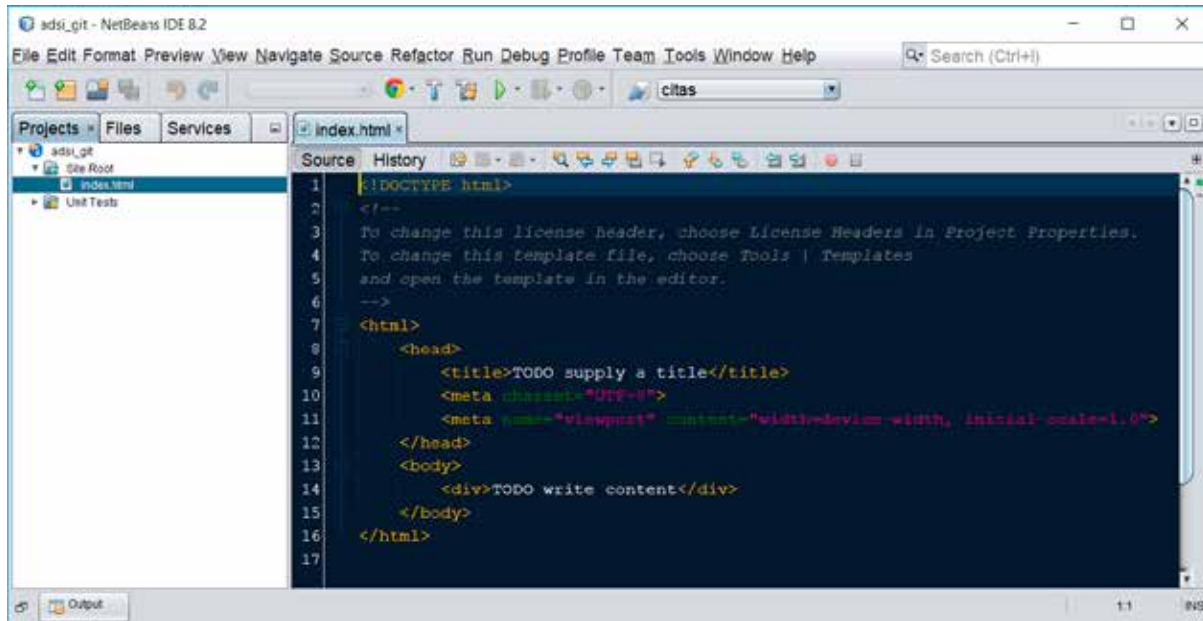


Figura 5.6. Creación de un proyecto HTML5 en Netbeans parte 5.

Al ejecutar el proyecto por la opción “Run” u oprimiendo la tecla F6 aparece lo siguiente en pantalla:

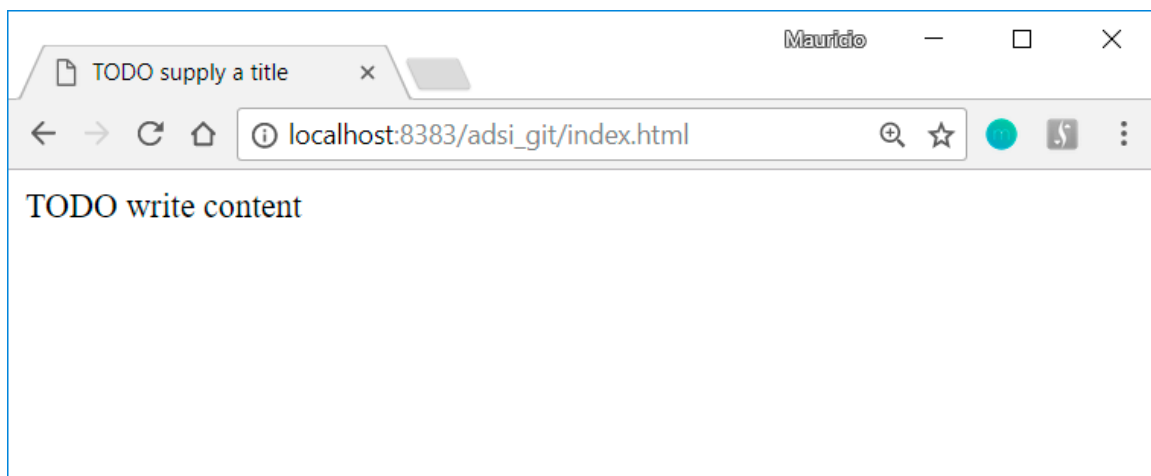


Figura 5.7. Creación de un proyecto HTML5 en Netbeans parte 6.



## 5.2. Creación del repositorio

Una vez creado el proyecto se inicia el repositorio con la opción del menú: “Team” → “Git” → “Initialize repository” como se muestra a continuación:

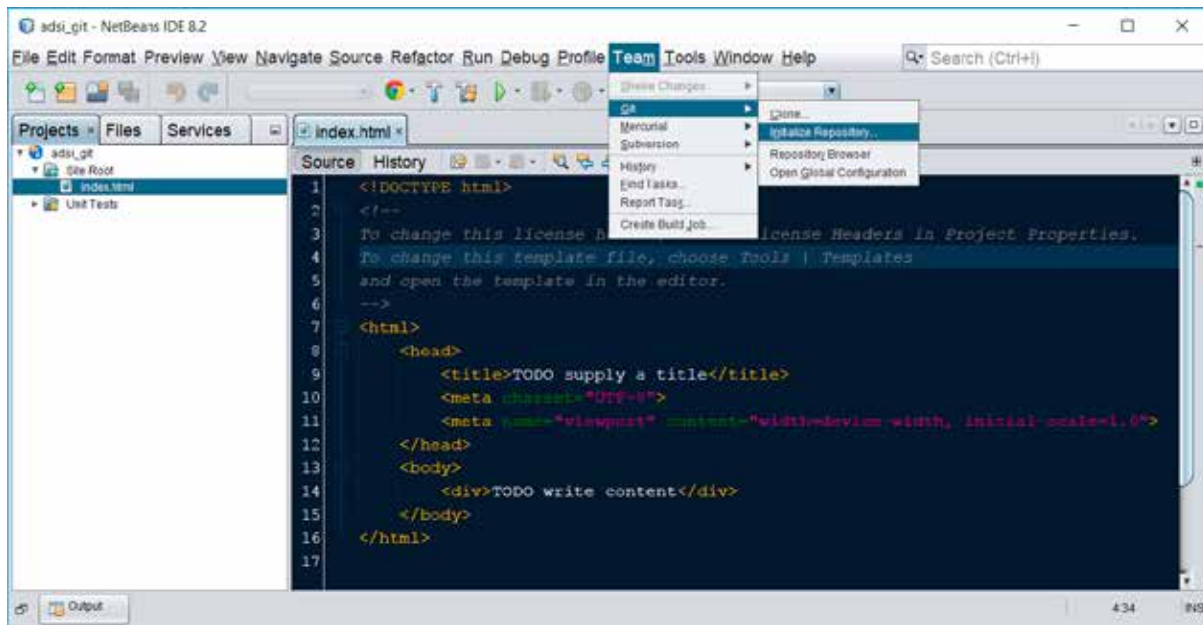


Figura 5.8. Iniciación del repositorio.

Una vez seleccionada la opción aparece lo siguiente en pantalla:

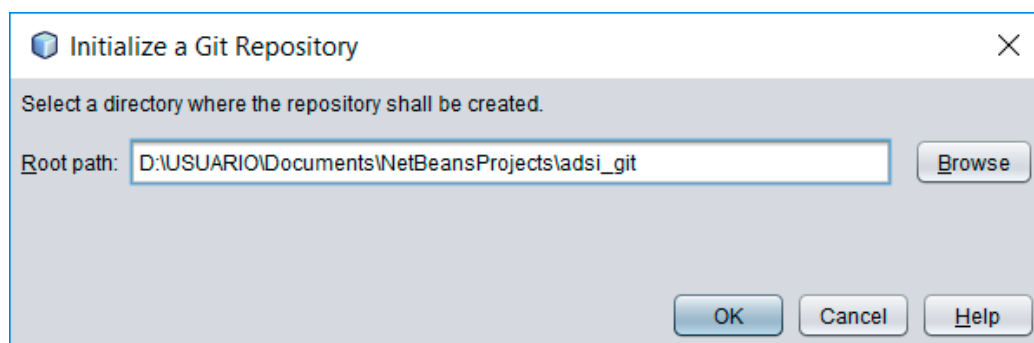


Figura 5.9. Iniciación del repositorio parte 2.

En este punto Netbeans indica el nombre del repositorio que por defecto es el mismo nombre del proyecto. Se debe oprimir el botón “Ok” para continuar. Aparecerá lo siguiente en la consola “Output”:

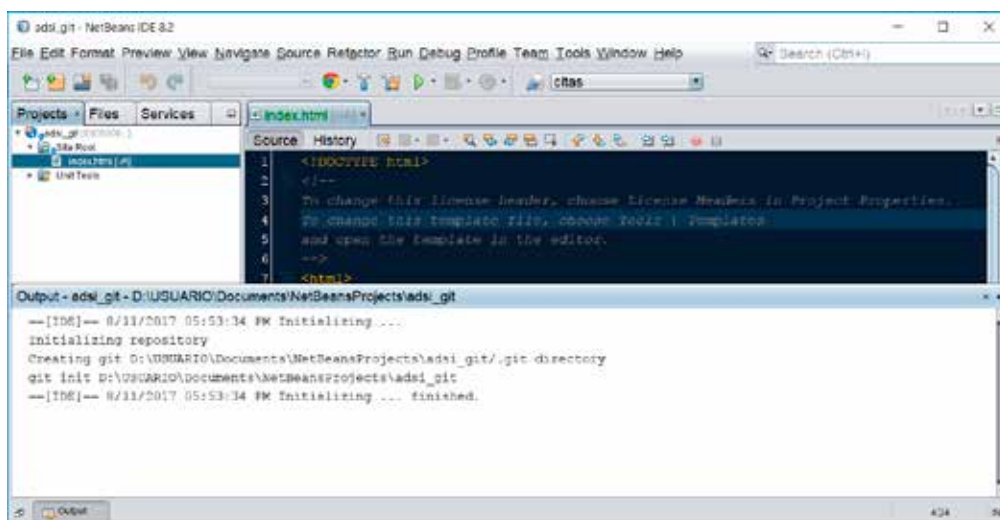


Figura 5.10. Iniciación del repositorio parte 3.

La ventana “Output” muestra los mensajes. En ellos se puede ver que el comando aplicado fue “git init” y el nombre del directorio solicitado. Por último, muestra que finalizó sin mostrar ningún error.

Se puede observar también que la pestaña que muestra el nombre del archivo “index.html” se encuentra de color verde. Lo anterior indica que el archivo está siendo controlado por Git.

## 5.3. Primera confirmación o commit

Se va a tomar este código como la versión uno del proyecto. Para indicar esto a Git se deben agregar los archivos con el comando “git add” que en esta versión integrada de Git se encuentra en “Team” → “Add” como se muestra a continuación:

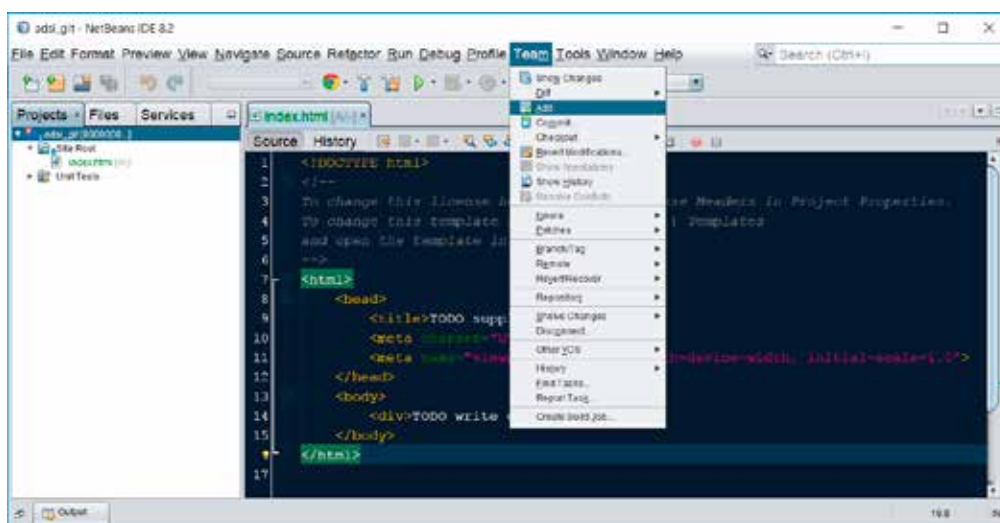


Figura 5.11. Adición de archivos al repositorio.

Una vez ejecutada la opción se puede observar en la ventana “output” los comandos Git usados por netbeans.

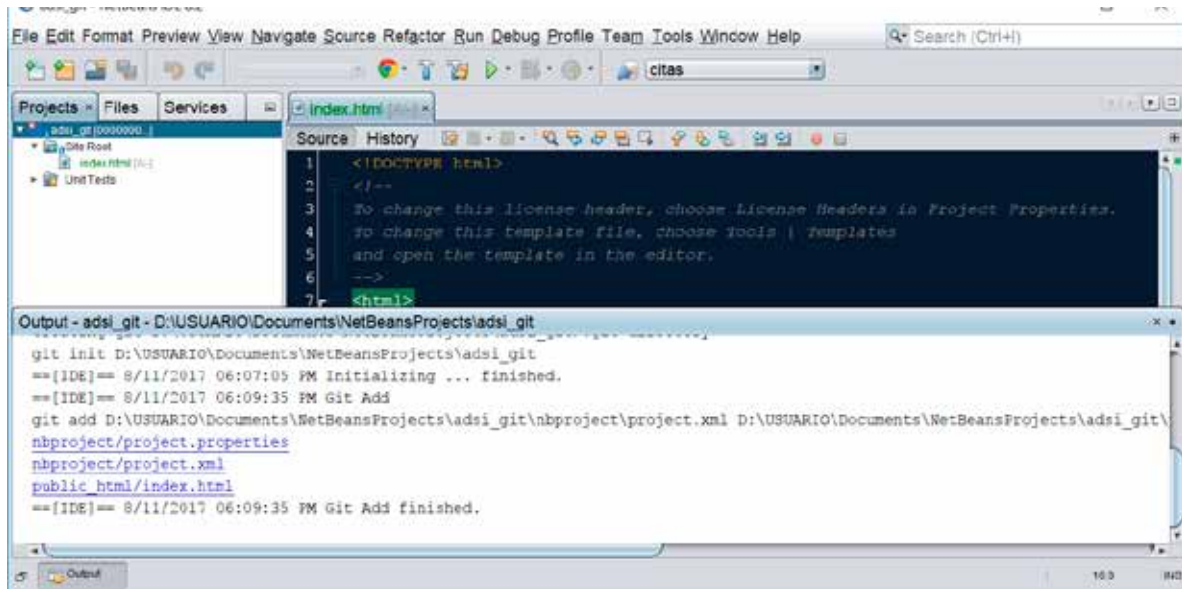


Figura 5.12 Adición de archivos al repositorio parte 2.

Se puede observar que el comando Git utilizado por Netbeans fue: “git add “. Este comando tomó como parámetros además del archivo “Index.html” otros usados internamente por Netbeans.

Luego se procede a confirmar los cambios. Para lo anterior se usa la opción del menú: “Team” → “Commit ...” aparece lo siguiente en pantalla:

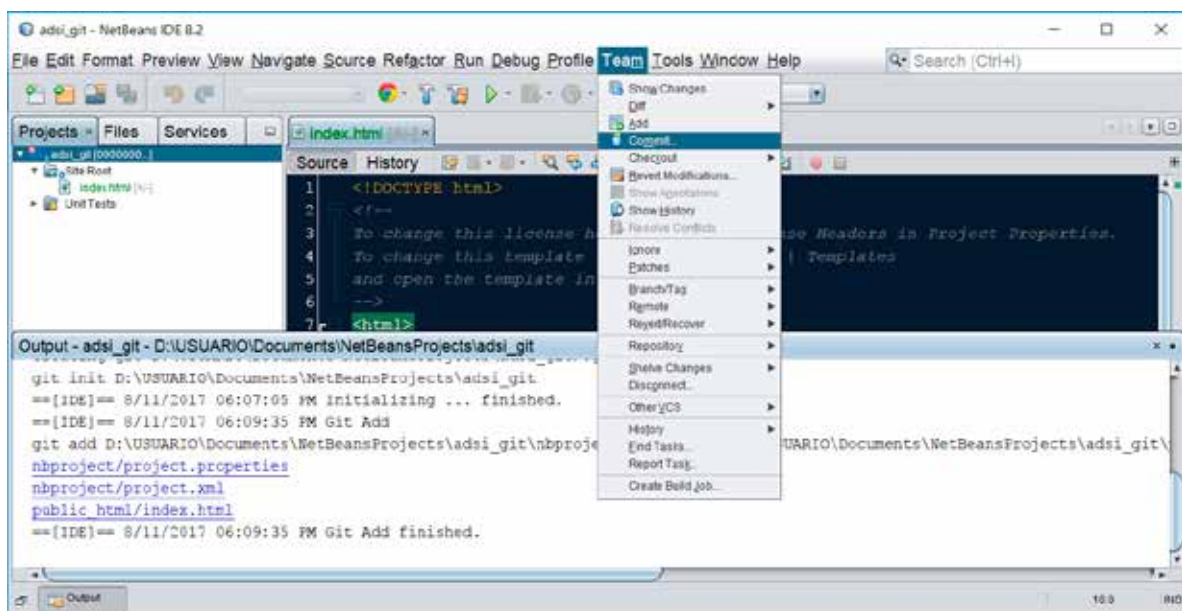


Figura 5.13 Primera confirmación de cambios o commit del proyecto.

Una vez seleccionada la opción aparece lo siguiente:

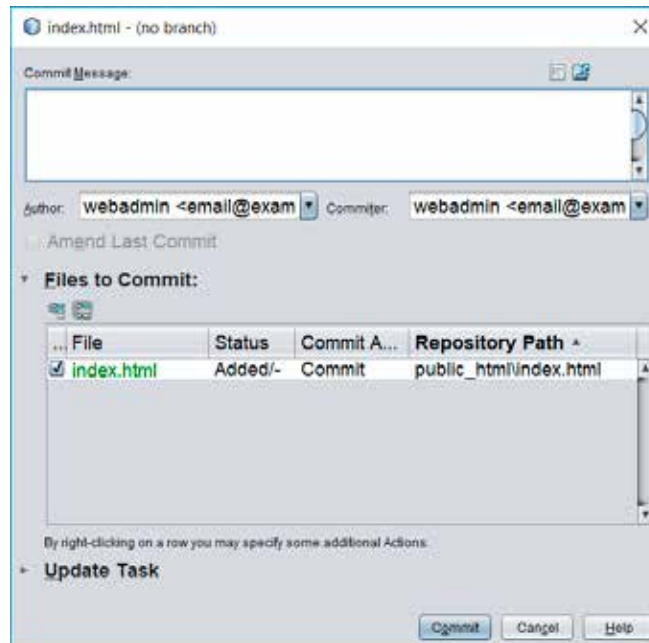


Figura 5.14 Primera confirmación de cambios o commit del proyecto parte 2.

En el espacio para el “Commit message” se ingresa el nombre de la primera versión que para este ejemplo se usará “versión\_1.0” así:

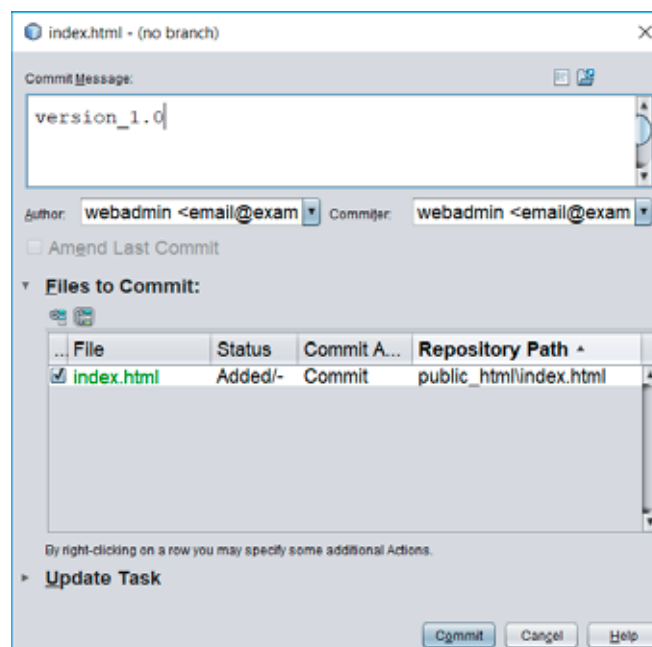
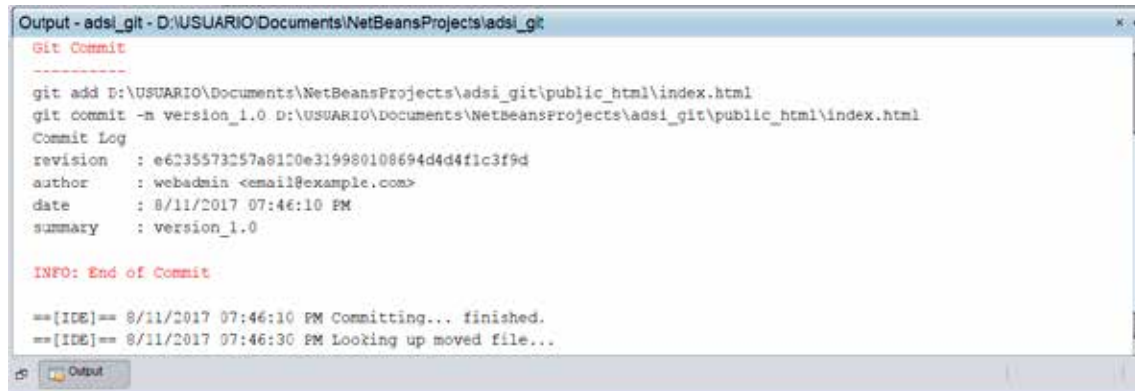


Figura 5.15. Primera confirmación de cambios o commit del proyecto parte 3.

Por último, se oprime el botón “Commit”. Aparece lo siguiente en la ventana “Output” del editor:



```

Output - adsi_git - D:\USUARIO\Documents\NetBeansProjects\ads_i_git
Git Commit
-----
git add D:\USUARIO\Documents\NetBeansProjects\ads_i_git\public_html\index.html
git commit -m version_1.0 D:\USUARIO\Documents\NetBeansProjects\ads_i_git\public_html\index.html
Commit Log
revision : e6235573257a8120e319980108694d4d4f1c3f9d
author   : webadmin <email@example.com>
date     : 8/11/2017 07:46:10 PM
summary  : version_1.0

INFO: End of Commit

==[IDE]== 8/11/2017 07:46:10 PM Committing... finished.
==[IDE]== 8/11/2017 07:46:30 PM Looking up moved file...
  
```

Figura 5.16. Primera confirmación de cambios o commit del proyecto parte 4.

Se puede observar que Netbeans usó internamente el comando “git commit” y el resultado se muestra en la ventana “Output”. Además, se muestra el hash, la fecha y hora, el usuario y el autor.

Se puede observar que una vez confirmados los cambios el nombre de la pestaña del archivo “index.html” cambia del color verde al negro.

Se pueden ver los cambios que ha tenido el proyecto hasta el momento usando la opción “Team” → “History” → “Show history” así:

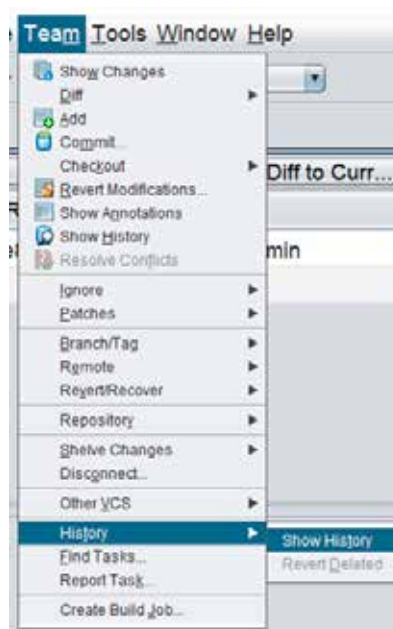


Figura 5.17.1. Primera confirmación de cambios o commit del proyecto parte 5.



Una vez seleccionada la opción aparece lo siguiente en pantalla:

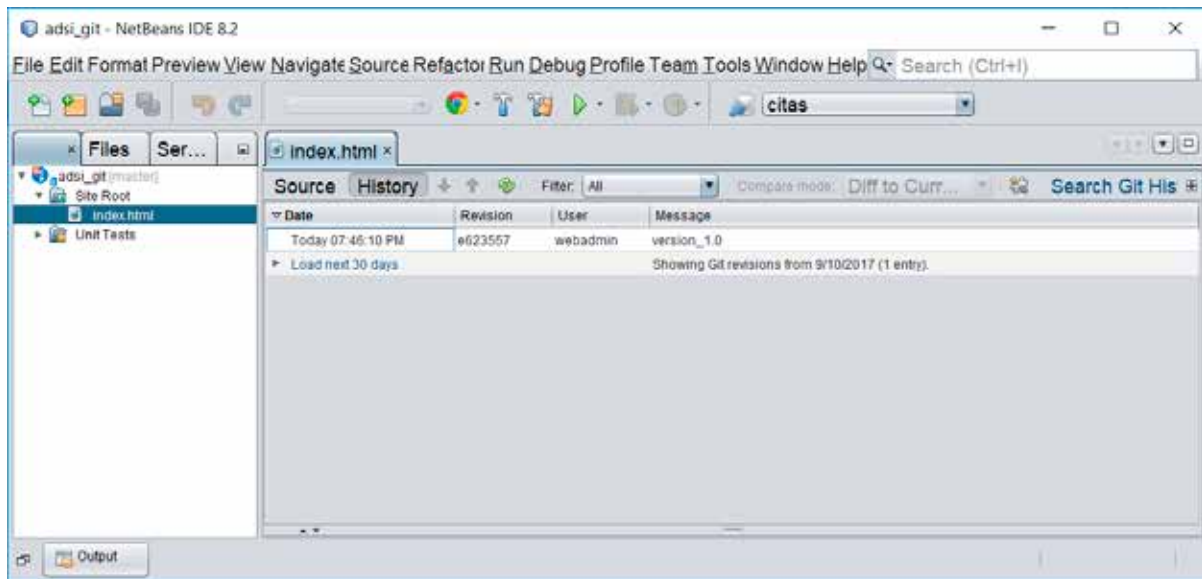


Figura 5.17.2. Primera confirmación de cambios o commit del proyecto parte 6.

Se puede observar que se relaciona el único cambio realizado cuyo mensaje es “version\_1.0”. También se puede observar que el historial es ahora otra pestaña del editor. Para regresar al código sólo hay que oprimir el nombre de la pestaña “Source”. Una vez realizada lo anterior aparece lo siguiente:

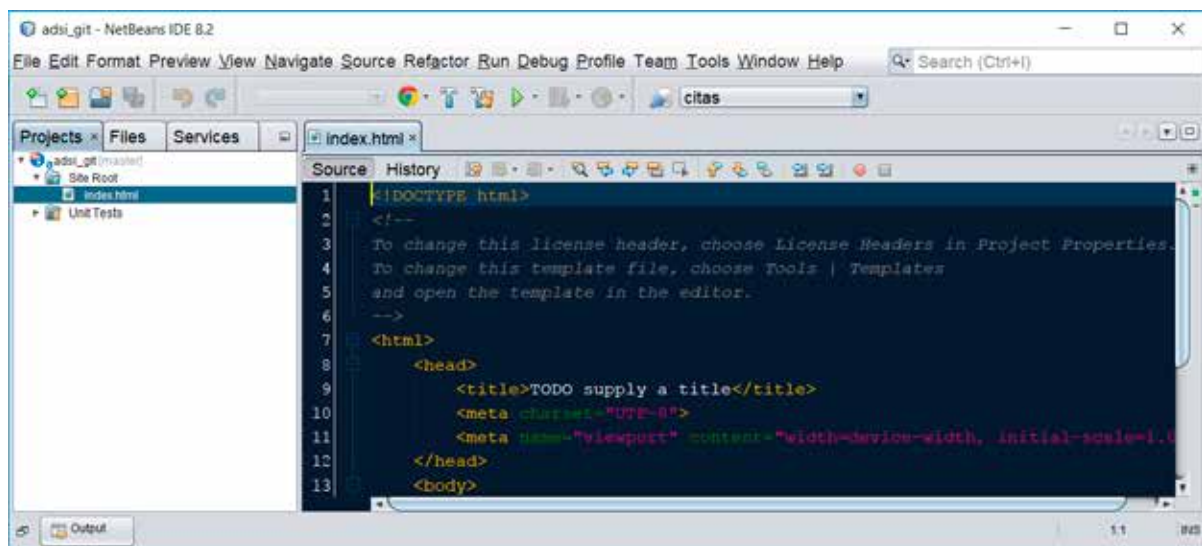


Figura 5.17.3. Primera confirmación de cambios o commit del proyecto parte 7.

## 5.4. Aplicación de nuevos cambios al proyecto

En el numeral anterior se realizó la confirmación de la primera versión del código fuente. A medida que un proyecto avanza se van introduciendo cambios los cuales se irán agregando usando la opción commit.

Se introducirá un cambio al archivo “index.html” que consistirá en la adición de una línea de código así:

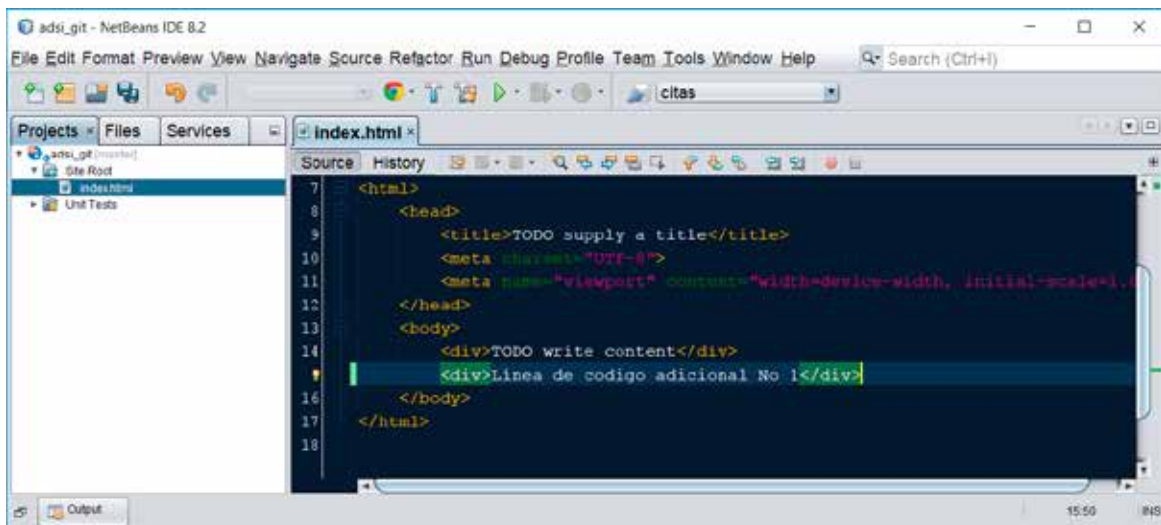


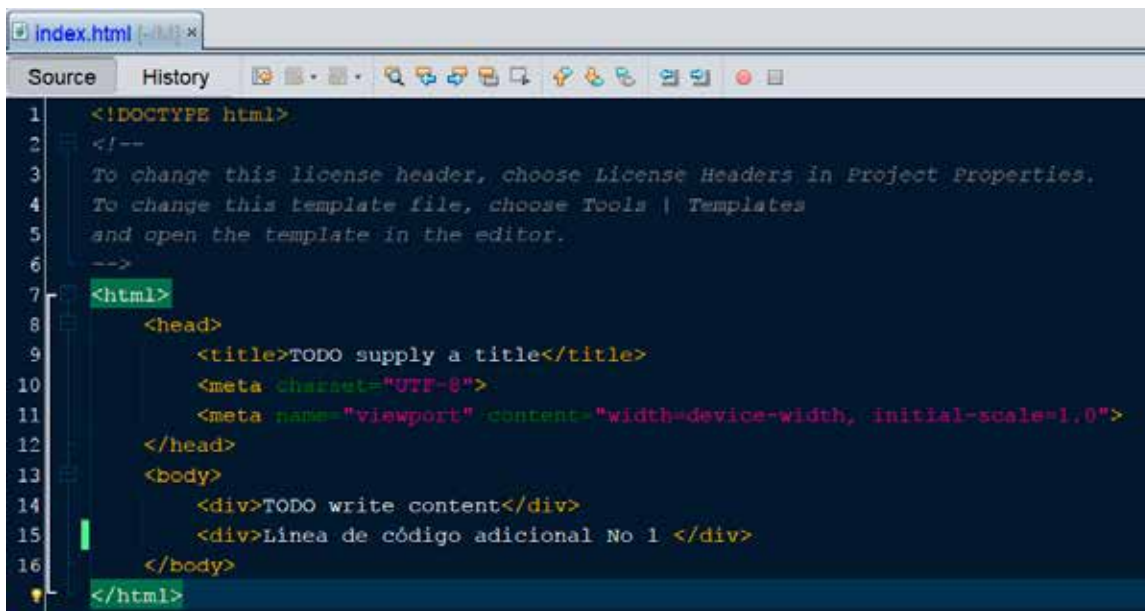
Figura 5.18. Adición de una línea de código al archivo index.html.

Luego se procede a agregar los cambios con la opción “Team” → “Add”. Aparece lo siguiente en la consola:

```
==[IDE]== 8/11/2017 06:29:54 PM Searching - D:\USUARIO\Documents\NetBeansProjects\adsi_git... finished.
==[IDE]== 8/11/2017 06:52:22 PM Git Add
==[IDE]== 8/11/2017 06:52:22 PM Git Add finished.
```



En este punto se puede observar que el color de la pestaña ha cambiado a azul y tiene un indicador “-M” que indica que el archivo ha sido modificado.



```
1 <!DOCTYPE html>
2 <!--
3 To change this license header, choose License Headers in Project Properties.
4 To change this template file, choose Tools | Templates
5 and open the template in the editor.
6 -->
7 <html>
8 <head>
9 <title>TODO supply a title</title>
10 <meta charset="UTF-8">
11 <meta name="viewport" content="width=device-width, initial-scale=1.0">
12 </head>
13 <body>
14 <div>TODO write content</div>
15 <div>Línea de código adicional No 1 </div>
16 </body>
</html>
```

Figura 5.19 Adición de una línea de código al archivo index.html parte 2.

Por último se genera una nueva confirmación que se llamará: “version\_1.1” así:

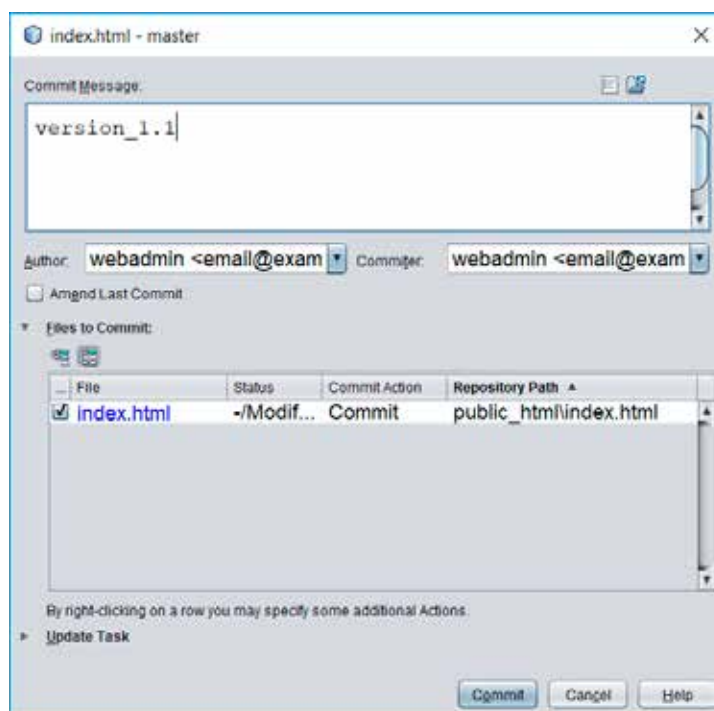


Figura 5.20. Adición de una línea de código al archivo index.html parte 3.



Una vez realizada se oprime el botón “Commit” aparece lo siguiente en la consola:

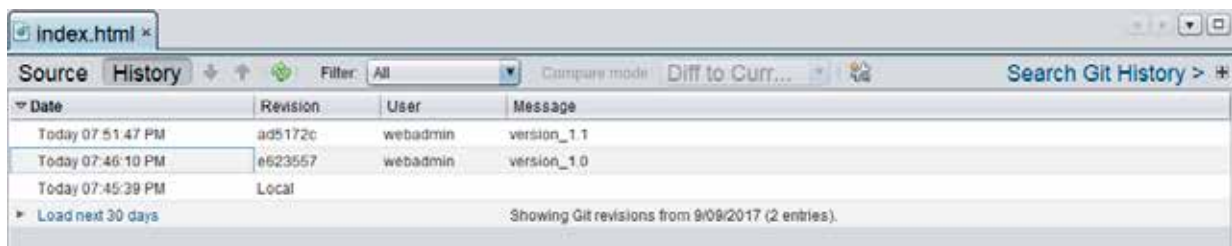


```

Output - adsi_git - D:\USUARIO\Documents\NetBeansProjects\adsi_git
Git Commit
git add D:\USUARIO\Documents\NetBeansProjects\adsi_git\public_html\index.html
git commit -m version_1.1 D:\USUARIO\Documents\NetBeansProjects\adsi_git\public_html\index.html
Commit Log
revision : ad5172c539f5977b779f170634b5fdb501c4581
author : webadmin <email@example.com>
date : 8/11/2017 07:51:47 PM
summary : version_1.1
INFO: End of Commit
  
```

Figura 5.21.1. Adición de una línea de código al archivo index.html parte 4.

En este punto se puede revisar el historial de cambios dando clic sobre la pestaña “History”. Aparece lo siguiente en pantalla:



| Date              | Revision | User     | Message     |
|-------------------|----------|----------|-------------|
| Today 07:51:47 PM | ad5172c  | webadmin | version_1.1 |
| Today 07:46:10 PM | e623557  | webadmin | version_1.0 |
| Today 07:45:39 PM | Local    |          |             |

Showing Git revisions from 9/09/2017 (2 entries).

Figura 5.21.2. Adición de una línea de código al archivo index.html parte 5.

## 5.5. Revisión de los cambios aplicados

Para ubicar el proyecto en una versión específica se utiliza la opción “Checkout revision” del menú “Checkout” de la opción “Team” como se muestra a continuación:

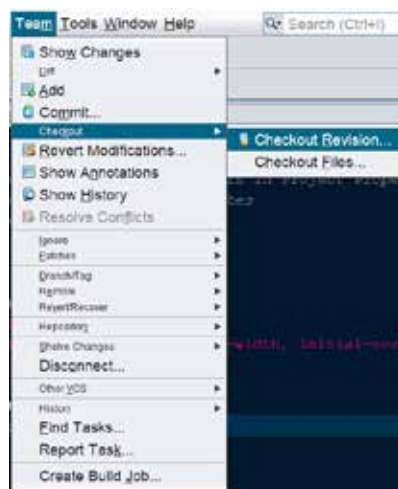


Figura 5.22. Revisión de cambios con el comando checkout.

Al seleccionar la opción aparece lo siguiente en pantalla:

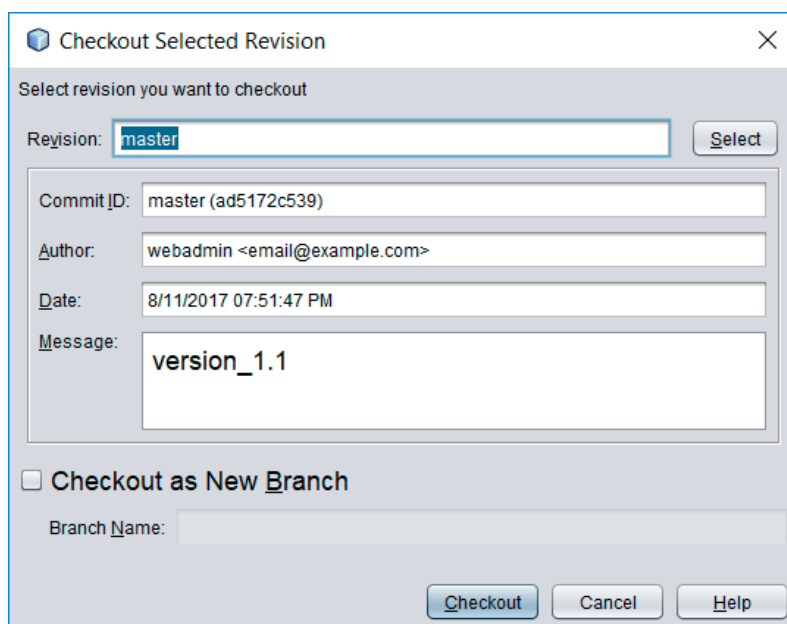


Figura 5.23. Revisión de cambios con el comando checkout parte 2.

La pantalla indica que está ubicado sobre la última confirmación o “master”. También pide que se seleccione el punto de recuperación o commit al cual se quiere revisar. Para realizar esta última selección se oprime el botón “Select”. Aparece lo siguiente en pantalla:

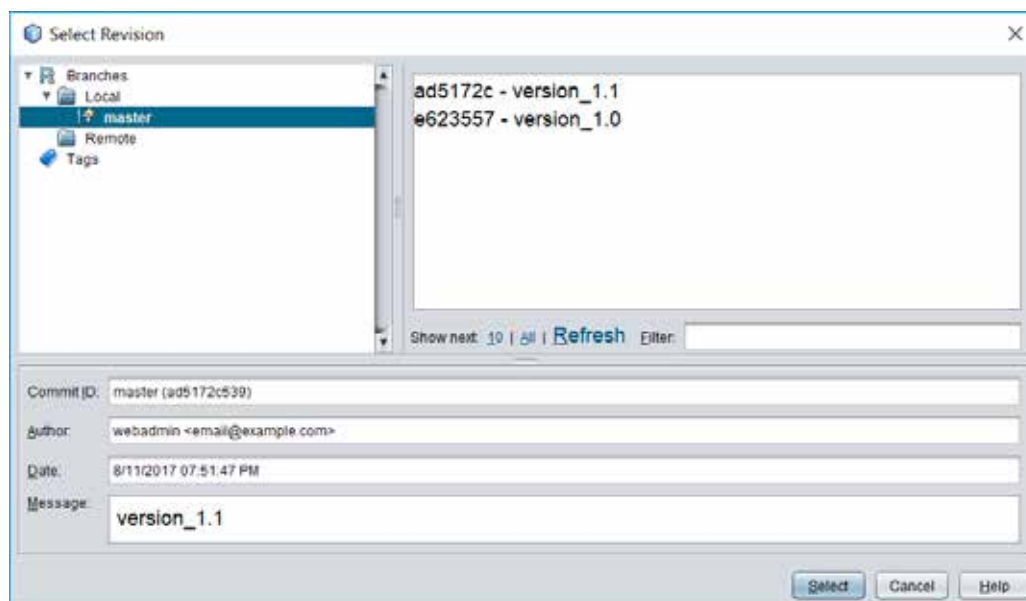


Figura 5.24. Revisión de cambios con el comando checkout parte 3.

Se puede observar que existen dos puntos de restauración o commits. Se solicitará que se regrese el proyecto al commit llamado “version\_1.0” que fue el commit inicial. Se selecciona este último dando clic sobre el nombre así:

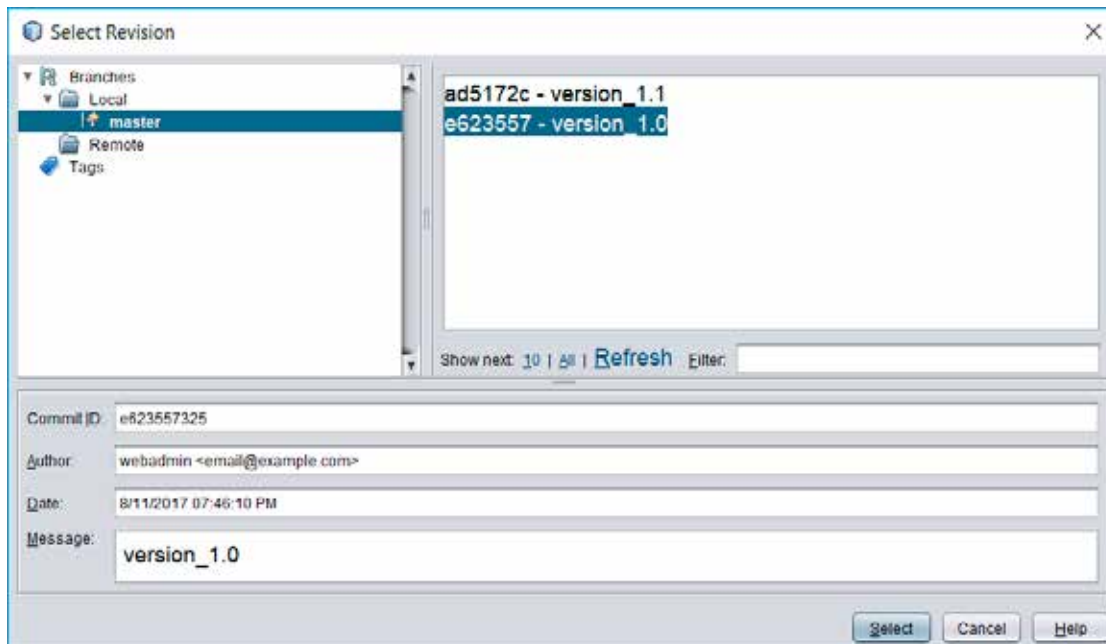


Figura 5.25. Revisión de cambios con el comando checkout parte 4.

Al dar clic el sistema actualiza los campos “Commit ID”, “Author”, “Date” y “Message”. Por último, se da clic sobre el botón “Select”. Aparece lo siguiente en pantalla:

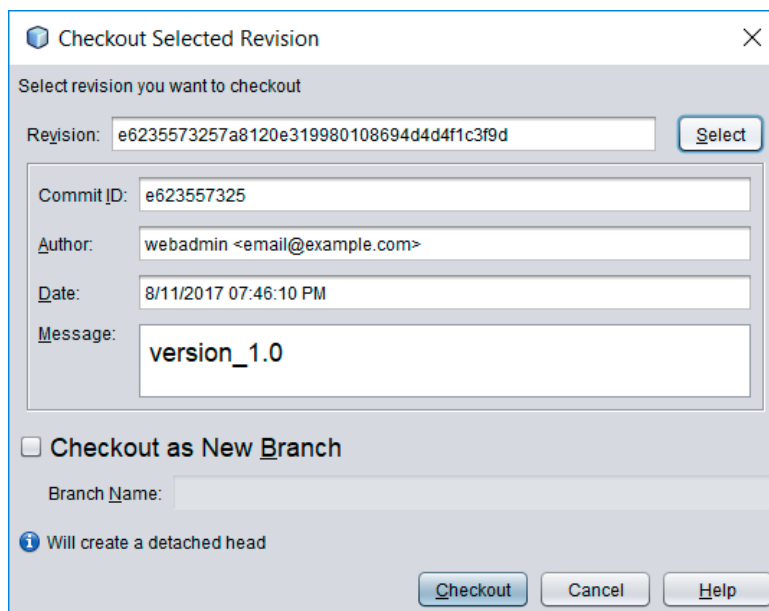


Figura 5.26. Revisión de cambios con el comando checkout parte 5.

La ventana es una última confirmación de los cambios. Para que la aplicación se ubique en la versión 1.0 se da clic sobre el botón “Checkout”. Una vez oprimido aparece lo siguiente en pantalla:

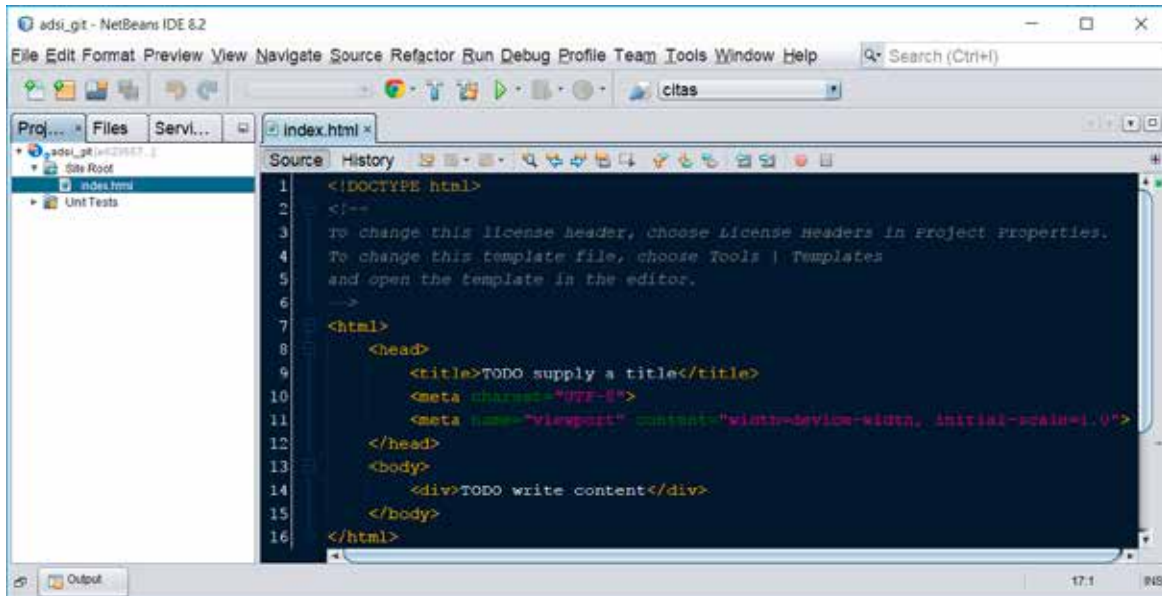


Figura 5.27. Revisión de cambios con el comando checkout parte 6.

Se puede observar que el archivo “index.html” volvió a su estado inicial. En este punto se puede revisar el proyecto como se encontraba en la versión 1.0. Una vez finalizada esta revisión se procede a regresar a la versión actual del mismo.

Para lo anterior se selecciona nuevamente la opción “Team” → “Checkout” → “Checkout revisions ...”. Luego al seleccionar el punto de restauración o commit se escoge “Master” como se muestra a continuación:

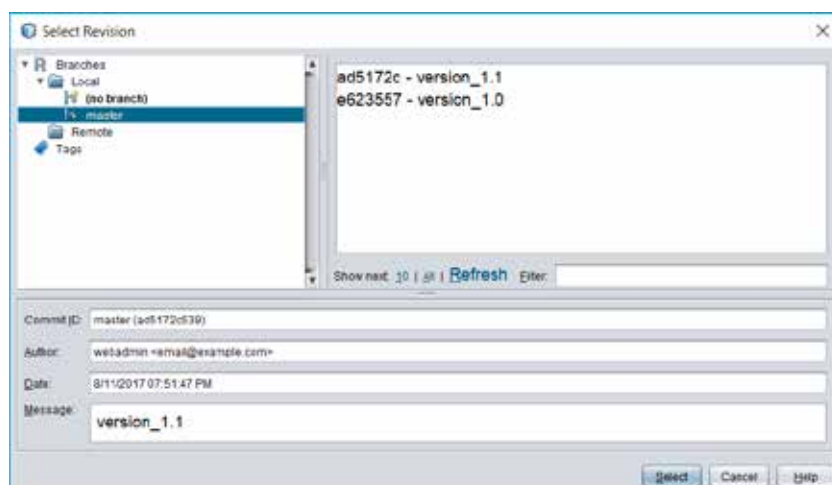


Figura 5.28. Revisión de cambios con el comando checkout parte 7.

Por último, se oprime el botón “Select”. Posteriormente se oprime el botón “Checkout” y el proyecto regresa a su estado actual como se muestra a continuación:

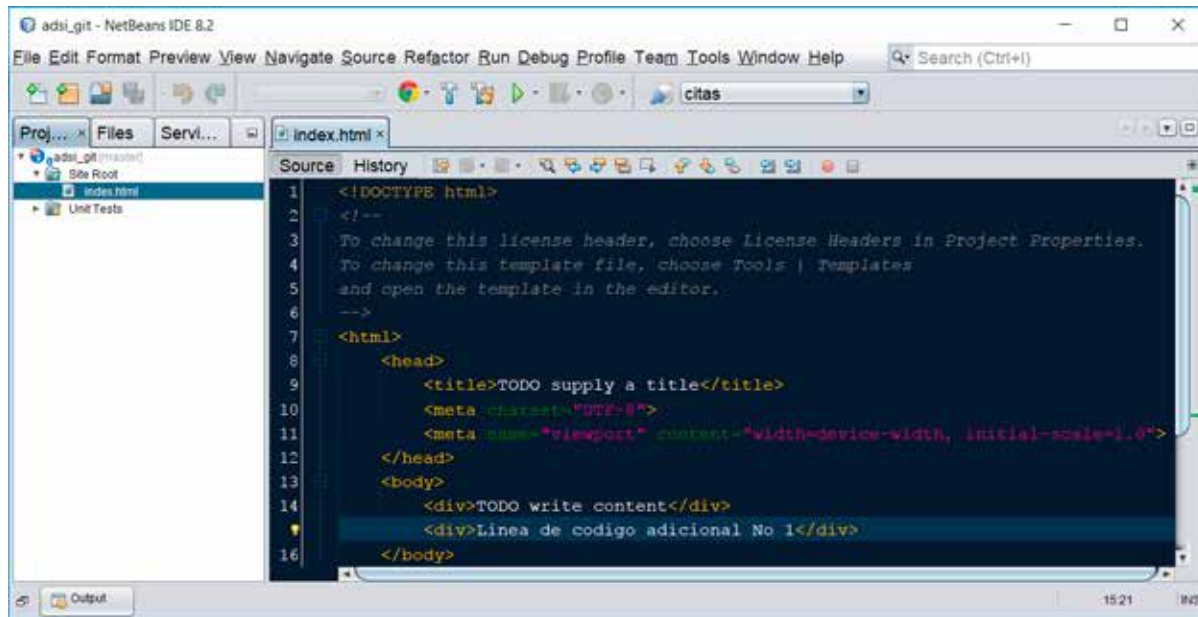


Figura 5.29. Revisión de cambios con el comando checkout parte 8.

## Glosario

**Hash:** firma o sello digital que permite identificar si los contenidos de uno o varios archivos han sido modificados.

**Linux:** sistema operativo de fuente abierta y gratuito de gran uso a nivel mundial.

**SHA-1:** algoritmo para la generación de identificadores de tipo Hash.

**Ssh:** acrónimo de Secure SHell. Software que permite encriptar las sesiones de trabajo.

**Vcs:** acrónimo de Version Control System. Sistema de control de versiones de software.

## Bibliografía

Chacon, S., Straub, B. (2014). *Pro Git Second Edition*. Apress Open.

## Control del documento

### CONSTRUCCIÓN OBJETO DE APRENDIZAJE



#### VERSIONAMIENTO DE CÓDIGO CON GIT

Centro Industrial de Mantenimiento Integral - CIMI  
Regional Santander

**Líder línea de producción:** Santiago Lozada Garcés

**Asesores pedagógicos:** Rosa Elvia Quintero Guasca  
Claudia Milena Hernández Naranjo

**Líder expertos temáticos:** Rita Rubiela Rincón Badillo

**Experto temático:** Nelson Mauricio Silva Maldonado

**Diseño multimedia:** Eulises Orduz Amezquita

**Programador:** Francisco José Lizcano Reyes

**Producción de audio:** Víctor Hugo Tabares Carreño

Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de la licencia que el trabajo original.



**creative  
commons**

Git © y el logotipo de Git son marcas registradas o marcas comerciales de Software Freedom Conservancy, Inc., sede corporativa del Proyecto Git, en los Estados Unidos y / o en otros países.



**Registered trademark**