



1º DAM/DAW Sistemas Informáticos

U3. Sistemas de control de versiones - Git

2 - Git. Conceptos. Fundamentos. Entorno gráfico



Git. SourceTree

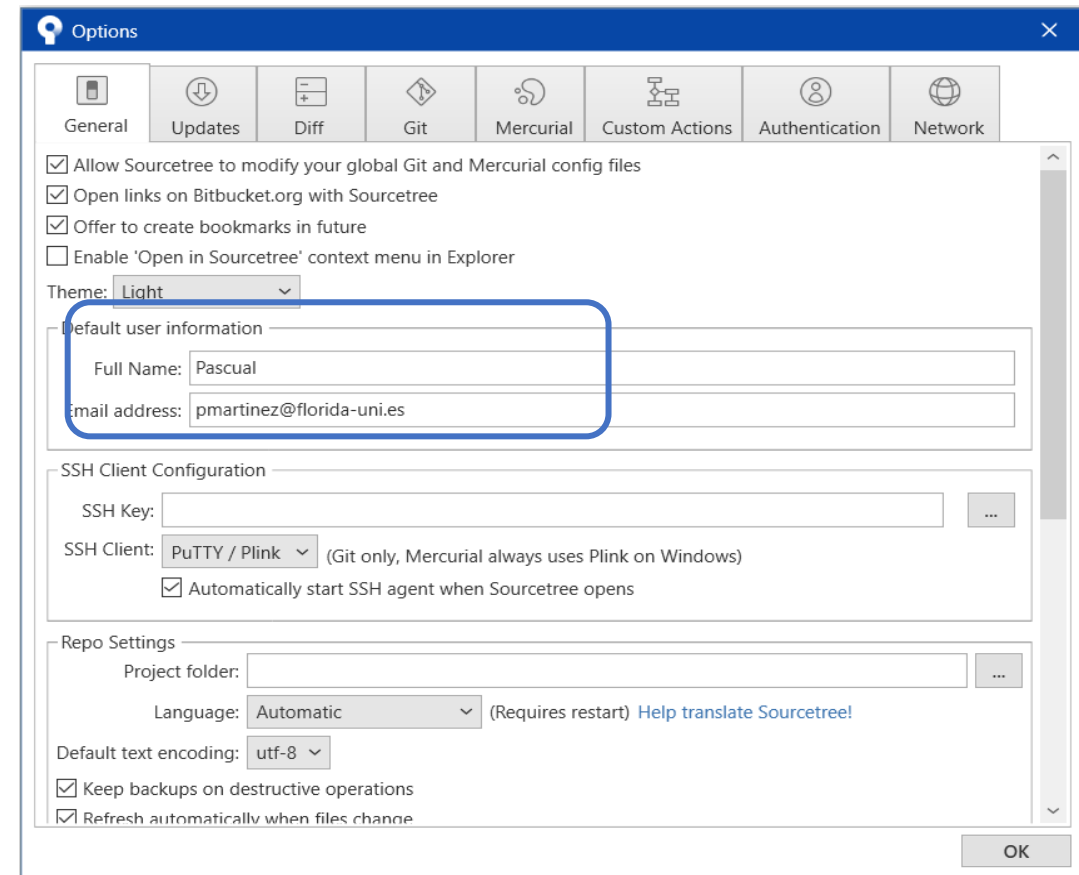
- Además de la consola (Git Bash), disponemos también de **entornos gráficos (GUI)** con los que poder **interactuar con Git**.
- Aunque son varias las distintas aplicaciones gráficas que podríamos utilizar, en nuestro caso, **usaremos SourceTree** de la compañía **Atlassian**. Cuando instalemos SourceTree, hay que tener en cuenta:

- Saltar la parte de Bitbucket, no lo usaremos.
- No instalar Mercurial, no lo usaremos.
- Si pide crear una cuenta Atlassian, lo haremos.



Git. SourceTree

- Una vez instalado **SourceTree**, podremos **confirmar que se han importado nuestros datos**, nombre y correo, configurados previamente mediante Git Bash y necesarios para poder firmar los cambios que se realicen en un repositorio.
- Pulsando **Tools > Options**, se nos abrirá la ventana de opciones:



Git. ¿Qué es un repositorio?

- **Según la R.A.E:** lugar donde se guarda algo. Del latín *repositorium*: armario, alacena.
- **En el ámbito de la informática:** es un **espacio centralizado** donde se almacena, organiza, mantiene y difunde **información digital**.
- En un proyecto de desarrollo de software **permite almacenar diferentes versiones del código y de la documentación**, a las que se podría acceder cuando se precise.



Diferencia entre directorio, carpeta y repositorio

- Un **directorio** es un elemento del S.O. que actúa como **almacén físico** de la última versión de los archivos y subdirectorios contenidos.
- Una **carpeta** es la **representación gráfica** de un directorio.
- Un **repositorio administra** las actualizaciones o modificaciones que se realizan sobre un conjunto de archivos y subdirectorios. Habitualmente enlazado con el directorio raíz de un proyecto.



Git. Iniciar repositorio

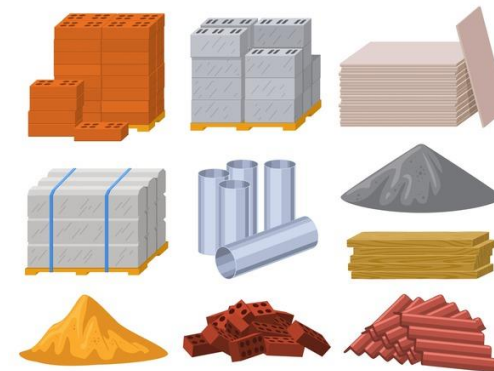
- Para trabajar con un VCS, en nuestro caso Git, lo primero que necesitamos **es iniciar un repositorio**. Podemos hacerlo de diferentes maneras:
 - **Clonación en local de un repositorio de Git ya existente**, por ejemplo, desde un servidor de repositorios (ejemplos: GitHub, GitLab, Bitbucket...) donde esté ubicado.
 - **Creación de un nuevo repositorio local de Git**.
- Independientemente de cómo iniciemos un repositorio, podremos configurarlo posteriormente para la **colaboración remota**.



Git. SourceTree. Iniciar repositorio local: Crear

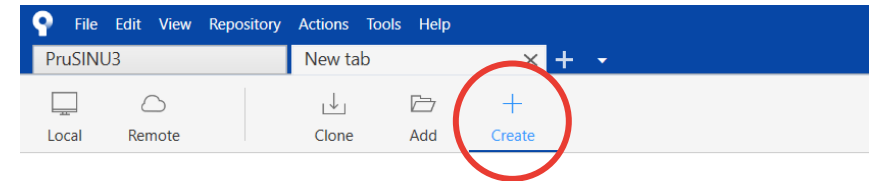
- **Creación de un repositorio:**

- Vamos a suponer ahora que hemos comenzado un proyecto de desarrollo de software desde cero. **Disponemos de un directorio con ficheros de código fuente y documentación en nuestro sistema de archivos local** y queremos crear un nuevo repositorio Git en base a este directorio.



Git. SourceTree. Iniciar repositorio local: Crear

- **Creación de un repositorio:**
 - Usaremos la opción “**Create**” de SourceTree.
 - Indicaremos:
 - La **ruta a la carpeta raíz del proyecto** en nuestro sistema de archivos.
 - Le pondremos un **nombre al repositorio** y pulsaremos el botón “Create”.



Create a repository

C:\Florida\Docencia\1 DAM Semi - SIN\Contenidos\U3\PruSINU3 Browse

PruSINU3

Git

☐ Create Repository On Account:

Create

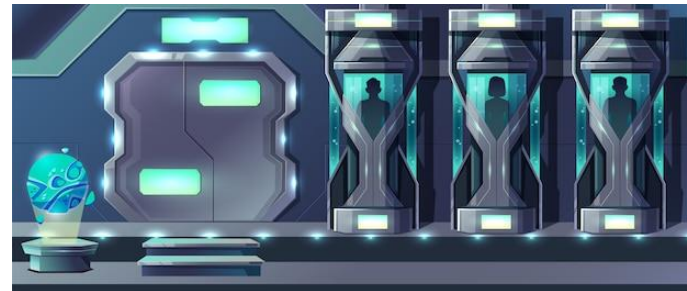
Git. SourceTree. Iniciar repositorio local: Clonar

- **Clonación de un repositorio:**

- Vamos a suponer que nos hemos incorporado a un proyecto de desarrollo de software ya existente.
- **El proyecto ya dispone de un repositorio Git remoto en un servidor y queremos clonar el repositorio remoto en nuestro sistema de archivos local.**

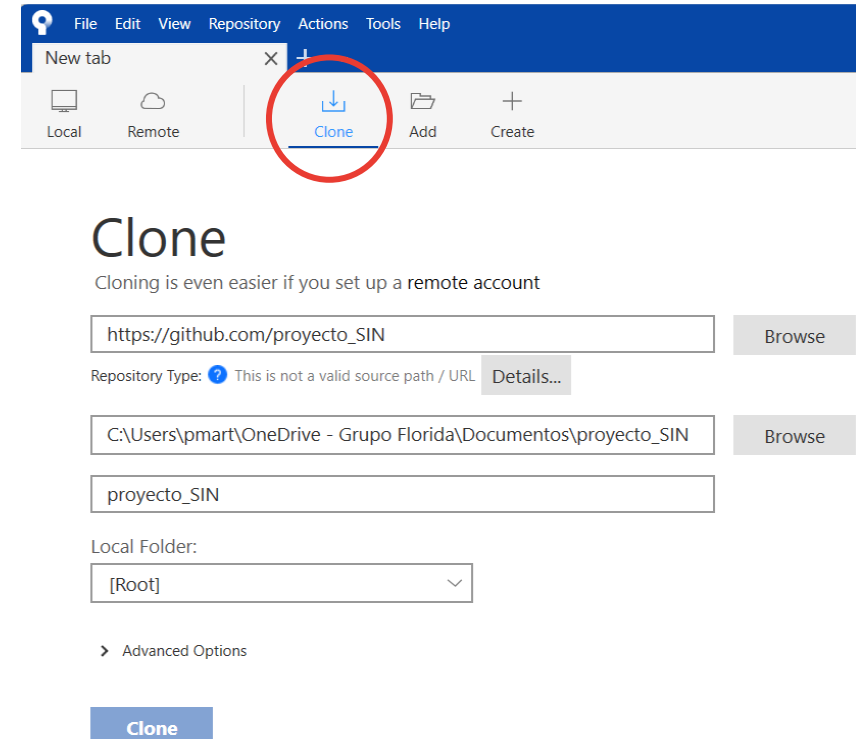


CLONE



Git. SourceTree. Iniciar repositorio local: Clonar

- **Clonación de un repositorio:**
 - Usaremos la opción “**Clone**” de SourceTree.
 - Indicaremos:
 - La **dirección del repositorio remoto**.
 - **Ruta** donde queremos ubicar el repositorio **en nuestro sistema de archivos** y pulsaremos el botón “Clone”.



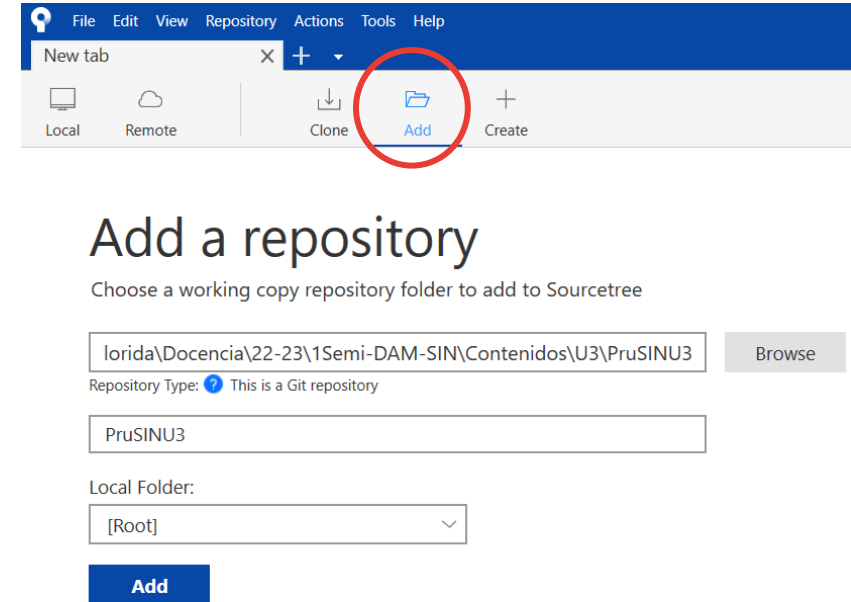
Git. SourceTree. Iniciar repositorio local: Añadir

- **Añadir un repositorio:**
 - Vamos a suponer ahora que **el proyecto ya dispone de un repositorio Git de ficheros. Disponemos de una copia del repositorio en nuestro sistema de archivos, o accesible por red, y vamos a comenzar a usar SourceTree para gestionarlo.**



Git. SourceTree. Iniciar repositorio local: Añadir

- **Añadir un repositorio:**
 - Usaremos la opción “**Add**” de SourceTree.
 - Indicaremos:
 - La **ruta a la carpeta raíz del proyecto** en nuestro sistema de archivos.
 - Le pondremos un **nombre al repositorio** y pulsaremos el botón “Add”.



Git. Consola. Iniciar repositorio local

- Cada una de las acciones que se realice mediante la interfaz gráfica tiene su equivalente a nivel de línea de comandos en la consola Git Bash.



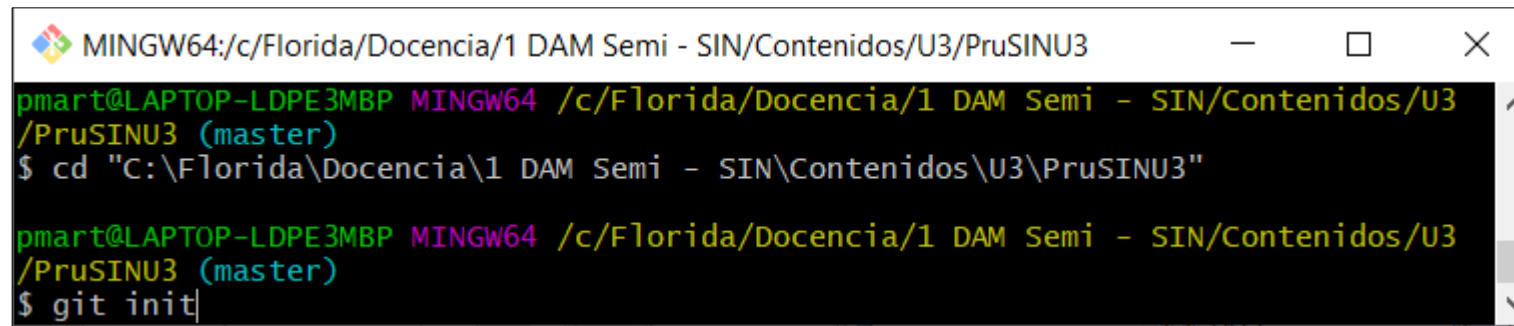
Git. Consola. Iniciar repositorio local

- En este caso, ubicados en la carpeta raíz del proyecto, podemos **añadir o crear un repositorio** mediante el comando **git init**:

\$ cd carpeta_raíz_proyecto

\$ git init

(alternativa: **\$ git init <carpeta_raíz_proyecto>**)



```
MINGW64:/c/Florida/Docencia/1 DAM Semi - SIN/Contenidos/U3/PruSINU3
pmart@LAPTOP-LDPE3MBP MINGW64 /c/Florida/Docencia/1 DAM Semi - SIN/Contenidos/U3
/PruSINU3 (master)
$ cd "C:\Florida\Docencia\1 DAM Semi - SIN\Contenidos\U3\PruSINU3"

pmart@LAPTOP-LDPE3MBP MINGW64 /c/Florida/Docencia/1 DAM Semi - SIN/Contenidos/U3
/PruSINU3 (master)
$ git init
```

Git. Consola. Iniciar repositorio local

- Para poder **clonar**, es decir, obtener una copia en local de **un repositorio Git ubicado en un servidor remoto**, lo haremos mediante el comando **git clone**:

\$ git clone URL_remota <carpeta_clonacion_local>

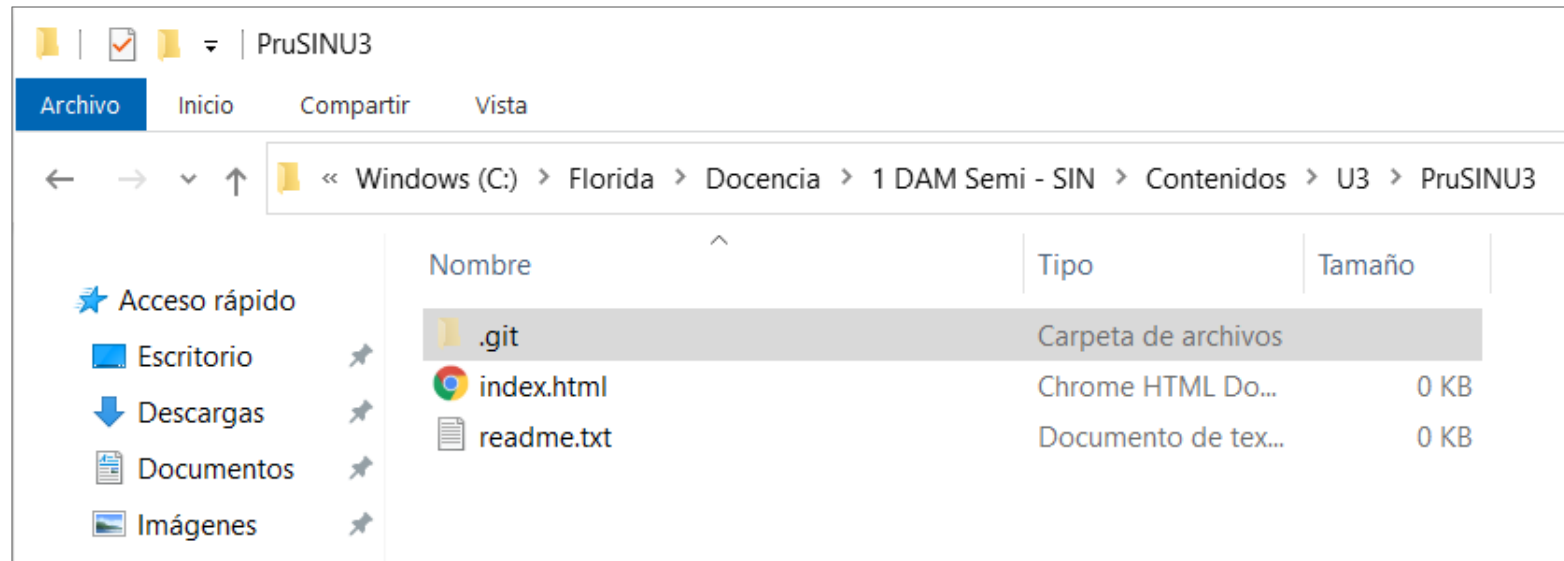


```
MINGW64:/c/Florida/Docencia/22-23/1Semi-DAM-SIN/Contenidos/U3/PruSIN...  
pmart@LAPTOP-LDPE3MBP MINGW64 ~  
$ cd "C:\Florida\Docencia\22-23\1Semi-DAM-SIN\Contenidos\U3\PruSINU3"  
pmart@LAPTOP-LDPE3MBP MINGW64 /c/Florida/Docencia/22-23/1Semi-DAM-SIN/Contenidos  
/U3/PruSINU3 (master)  
$ git clone https://github.com/libgit2/libgit2 clonacion_local
```

** En la ventana de ejemplo: si no indicáramos la “carpeta clonacion_local”, como parámetro opcional del comando, el sistema crearía una carpeta llamada “libgit2” en el sistema de archivos local.

Git. Carpeta “.git”

- En el momento iniciemos un repositorio de Git, ya sea a través de consola o mediante interfaz gráfica, Git creará una **carpeta oculta** llamada “.git” en la carpeta raíz de nuestro proyecto. En esta carpeta, Git guardará la información que usará para gestionar el repositorio.



Git. Los tres estados

- Una vez disponemos de un repositorio, una de las cuestiones más importantes en Git es conocer el **mecanismo de gestión de adiciones y modificaciones de archivos**, llamado **los tres estados**.



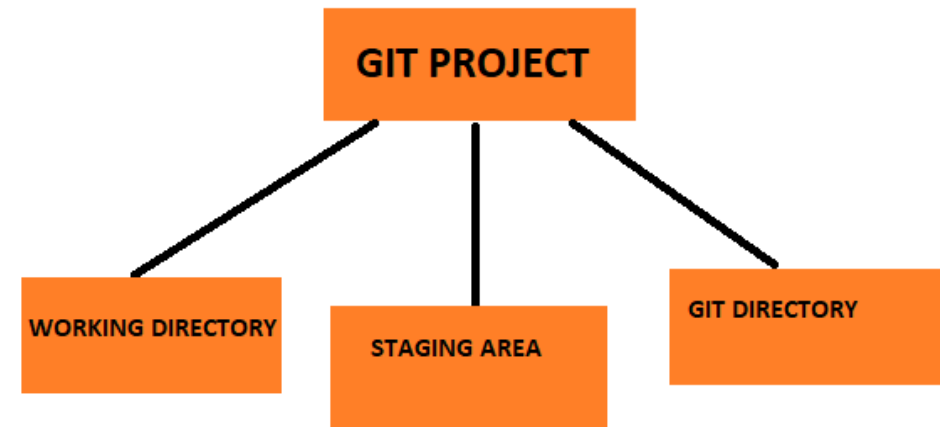
Git. Los tres estados

- Se basa en que **todos los archivos** de un repositorio Git se encuentran **en uno de estos tres estados**:
 - **Modificado (modified)**: un archivo ha sido **añadido o modificado**, pero los cambios todavía no se han consolidado , confirmado o actualizado en el repositorio.
 - **Preparado (staged)**: un archivo modificado ha sido **marcado para incluirlo** en la próxima consolidación, confirmación o actualización de cambios del repositorio.
 - **Confirmado (committed)**: un archivo preparado ha sido **consolidado**, confirmado actualizado o almacenado correctamente en el repositorio.

**** Consolidación, confirmación, actualización o almacenamiento son sinónimos en este contexto**

Git. Los tres estados. Secciones

- Un repositorio Git está formado por **tres secciones** :
 - El **directorio de trabajo (working directory)**.
 - El **área de preparación (staging area)**.
 - El **repositorio Git (repository)**.

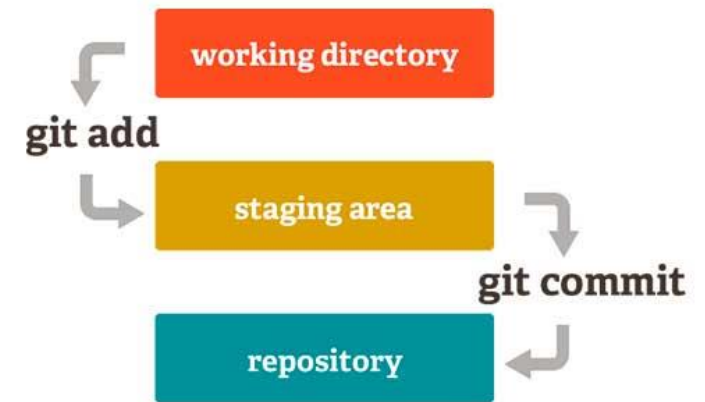


Git. Los tres estados. Secciones

- El **directorio de trabajo (working directory)**: es una copia en curso de una versión del proyecto. Está formado por los archivos modificados.
- El **área de preparación (staging area)**: almacena información acerca de lo que va a ir en la próxima confirmación o consolidación. Está formado por los archivos modificados y marcados para incluirlos en la próxima consolidación.
- El **repositorio Git (repository)**: almacena la base de datos de objetos para el proyecto sus metadatos. Es la parte más importante de Git. Está formado por los archivos confirmados.

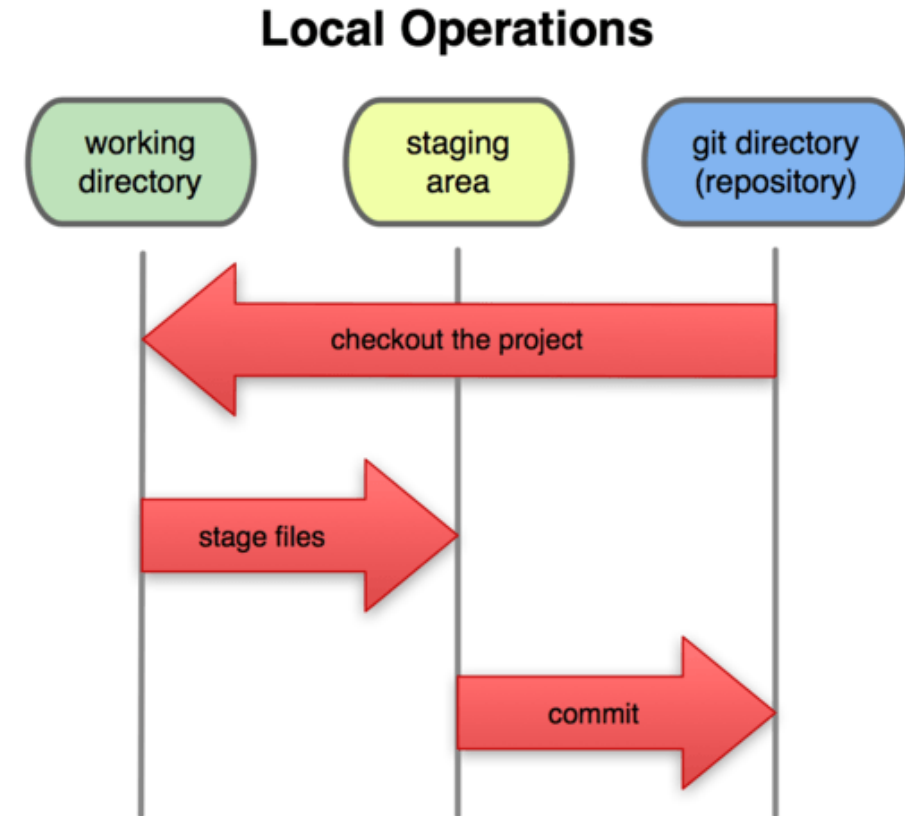
Git. Los tres estados

- Veamos ahora el **flujo de trabajo** en detalle:
 1. Cuando se produce un **cambio en el repositorio**, los contenidos afectados se sitúan en **el directorio del proyecto (working directory)**. Es decir, cuando se añade un archivo nuevo, o se modifica/elimina alguno existente.
 2. Los cambios se preparan para ser incluidos en la siguiente consolidación, **añadiéndolos (git add) al área de preparación (staging area)**.
 3. Cuando ejecutamos la **consolidación o confirmación (git commit)**, las actualizaciones del área de preparación pasarán a formar **parte del repositorio**.



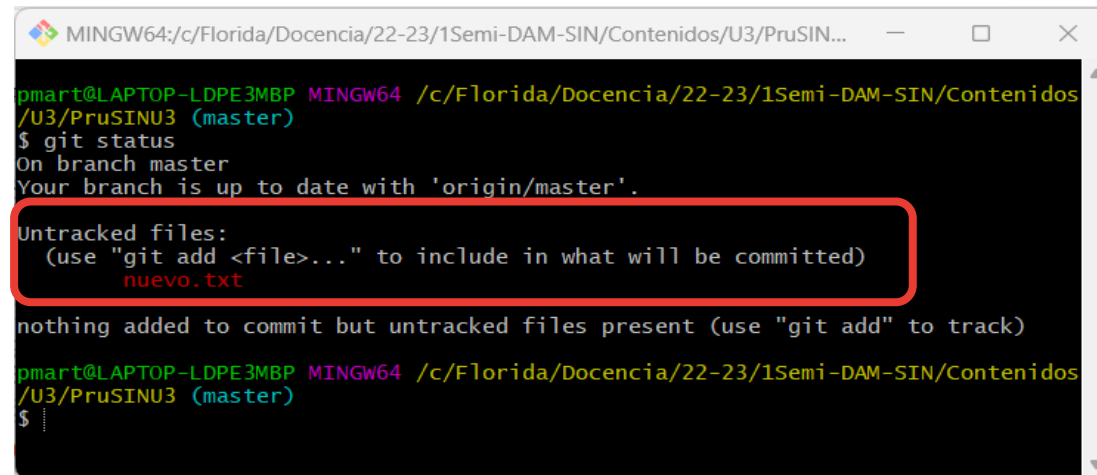
Git. Los tres estados

- Si una versión concreta de un archivo está en el directorio de Git, se considera confirmada (**committed**).
- Si se ha añadido al repositorio o ha sufrido cambios desde que se obtuvo del mismo, pero no se ha preparado, está modificado (**modified**), en el directorio de trabajo.
- Si ha sufrido cambios y ha sido añadida al área de preparación, está preparado (**staged**).



Git. Ciclo de vida del estado de los archivos

- Los archivos añadidos al repositorio, se denominan **archivos rastreados (tracked files)**, independientemente del estado en el que estén (consolidados, modificados o preparados).
- Los archivos nuevos que están en el directorio de trabajo y nunca habían pertenecido al repositorio, se denominan **archivos no rastreados (untracked files)**.

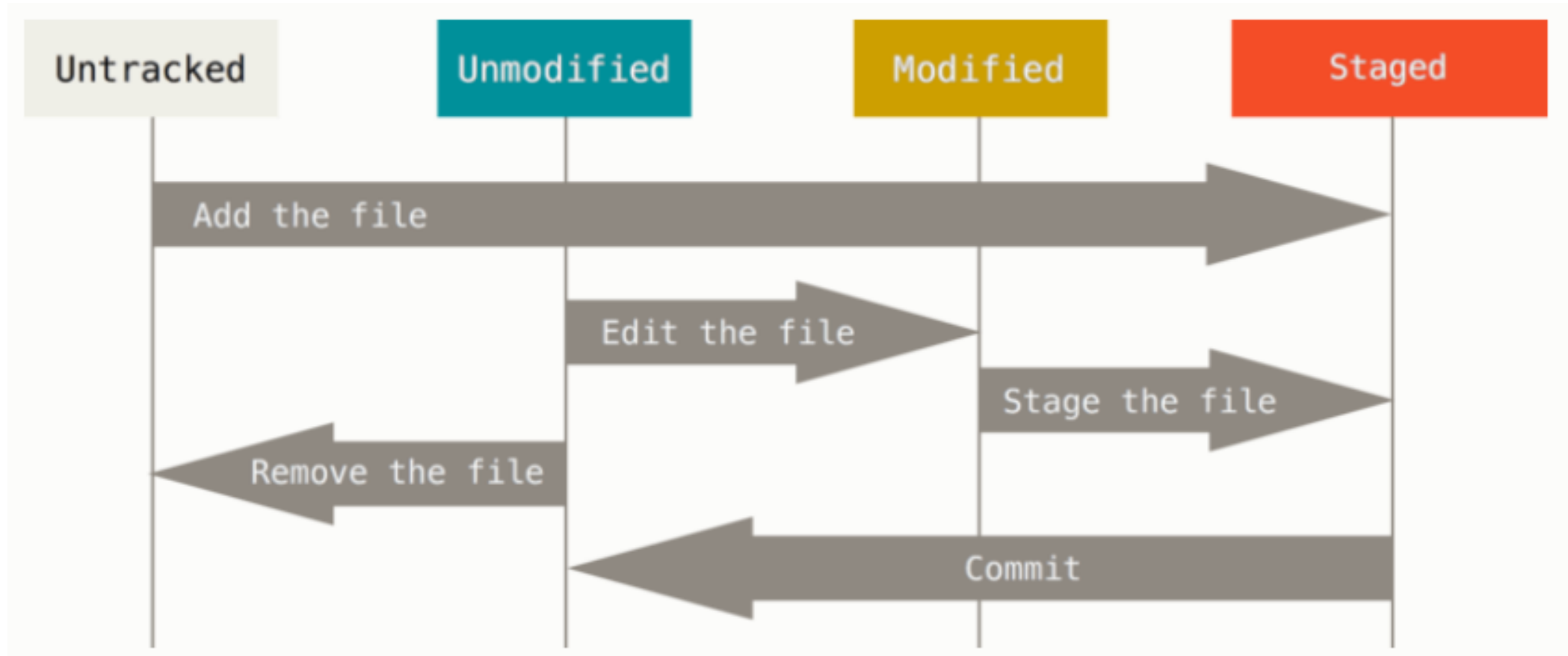


```
mingw64/c/Florida/Docencia/22-23/1Semi-DAM-SIN/Contenidos/U3/PruSIN...
pmart@LAPTOP-LDPE3MBP MINGW64 /c/Florida/Docencia/22-23/1Semi-DAM-SIN/Contenidos
/U3/PruSINU3 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

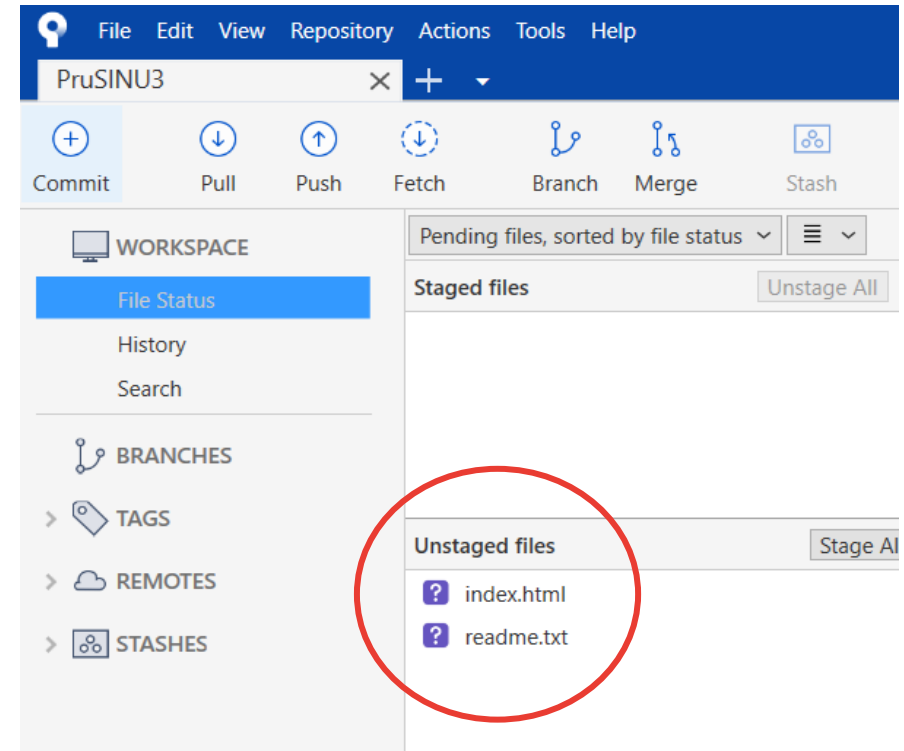
nothing added to commit but untracked files present (use "git add" to track)
pmart@LAPTOP-LDPE3MBP MINGW64 /c/Florida/Docencia/22-23/1Semi-DAM-SIN/Contenidos
/U3/PruSINU3 (master)
$
```

Git. Ciclo de vida del estado de los archivos



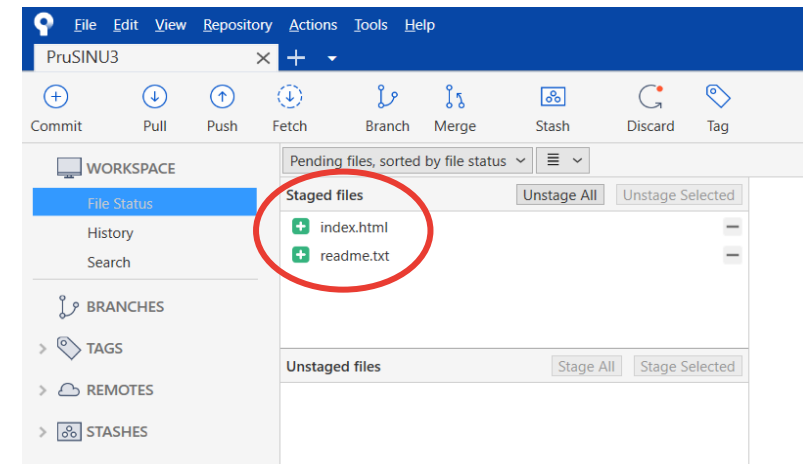
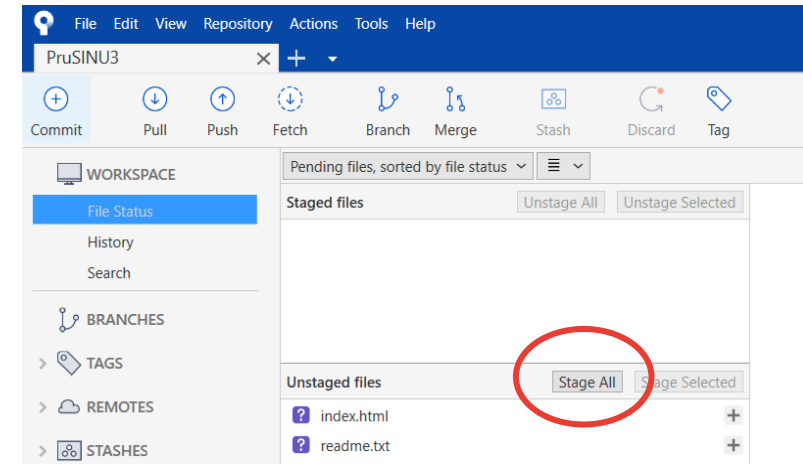
Git. SourceTree. Añadir cambios

- Una vez creado un repositorio, si por ejemplo se ha creado en base a una carpeta de nuestro sistema de archivos local, los ficheros que ya existían en la carpeta previamente aparecerán de forma automática, marcados como **pendientes de pasar al área de preparación (unstaged files)**. Es decir, aparecen como archivos en el directorio de trabajo.
- Sucederá lo mismo cuando añadamos o generemos un archivo nuevo en el directorio de trabajo.



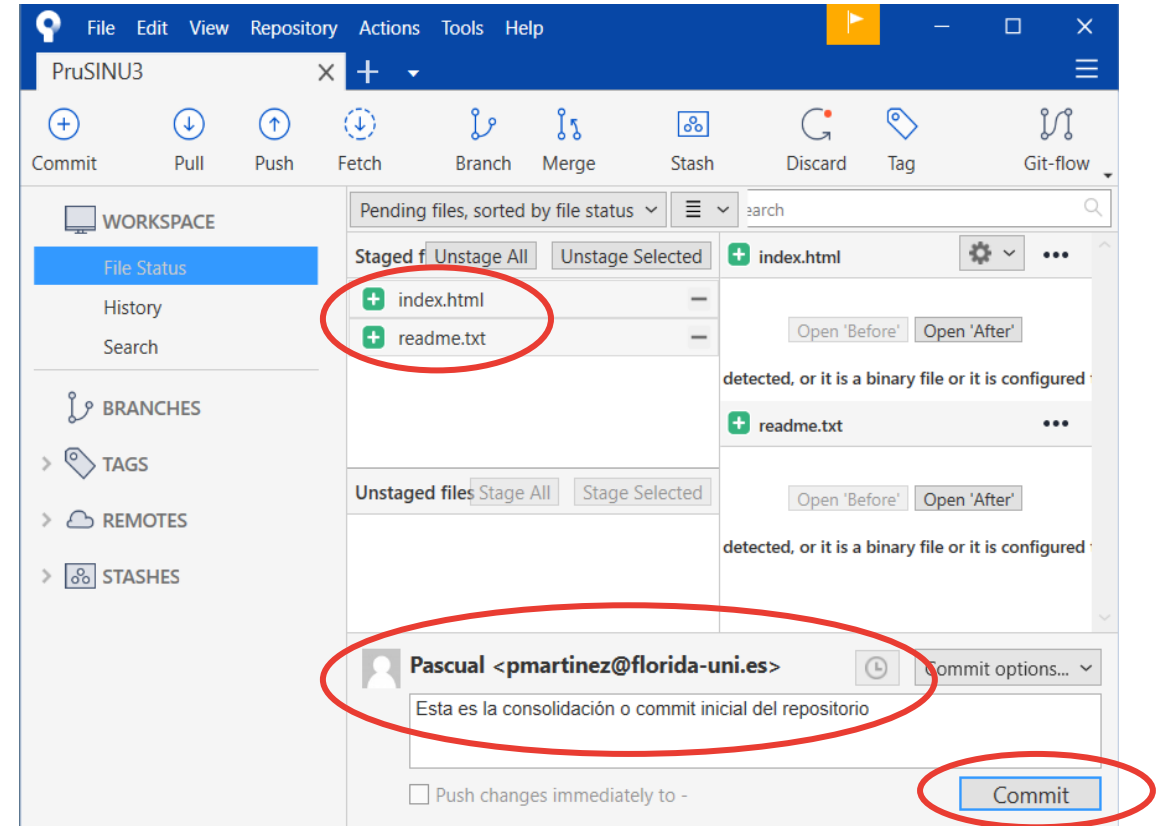
Git. SourceTree. Añadir cambios

- Pulsando el botón **“Stage All”**, pasaremos al área de preparación (staging area), tanto los ficheros nuevos no rastreados, como los modificados.
- Los ficheros ubicados en el área de preparación pasarán a formar parte del repositorio en el momento realicemos la acción de consolidación o confirmación (commit).



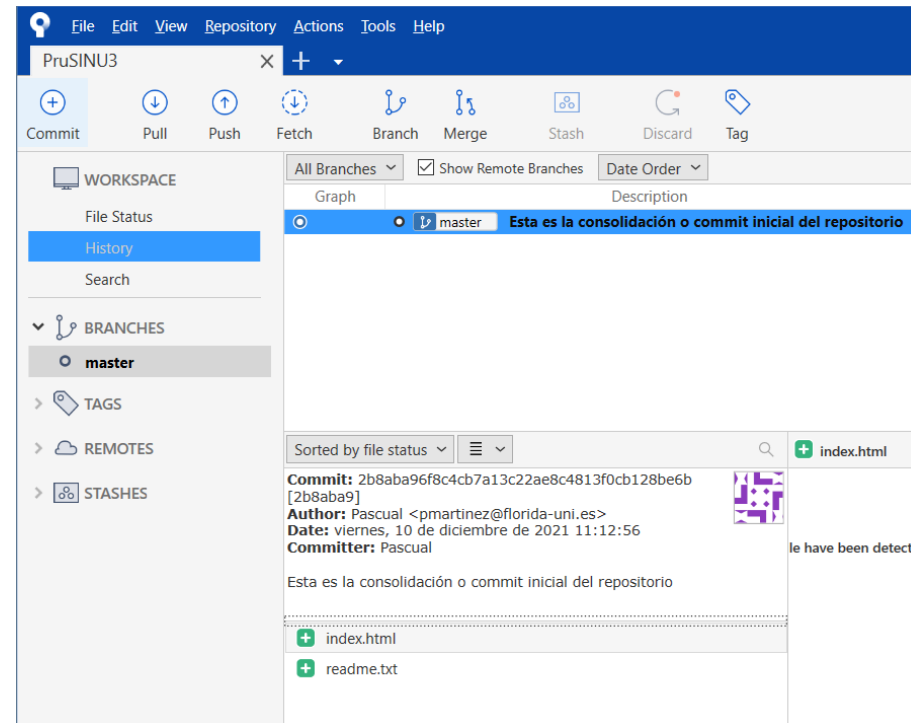
Git. SourceTree. Añadir cambios

- Para realizar una consolidación, confirmación, actualización o **commit**, podemos seguir los siguientes pasos:
 1. **Seleccionamos los ficheros** del área de preparación que queremos consolidar.
 2. **Introducimos un texto descriptivo** sobre la consolidación a realizar.
 3. **Pulsaremos el botón “Commit”**.



Git. SourceTree. Añadir archivos

- Una vez realizada la consolidación ya disponemos de un repositorio Git, con los ficheros de nuestro proyecto.



Git. Consola. Añadir cambios

- Estos pasos para consolidar los ficheros de nuestro proyecto en un repositorio Git, se podrían haber realizado también mediante comandos:
 - **\$ git add ruta_archivos:** añade archivos modificados o no rastreados al área de preparación.
 - **\$ git status:** nos muestra el estado de los ficheros:
 - **Modified en color rojo:** modificado y pendiente de añadir al área de preparación.
 - **Modified en color verde:** preparado y pendiente de consolidar en el repositorio.
 - **\$ git commit -m "Texto_descriptivo":** consolida en el repositorio los ficheros del área de preparación.

Git. Consola. Añadir cambios

- Con el comando “git status”, vemos un ejemplo de un archivo rastreado, modificado y pendiente de añadir al área de preparación.

“git status -s”: muestra información abreviada

```
MINGW64:/c/Florida/Docencia/1 DAM Semi - SIN/Contenidos/U3/PruSINU3
$ 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:   index.html

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

pmart@LAPTOP-LDPE3MBP MINGW64 /c/Florida/Docencia/1 DAM Semi - SIN/Contenidos/U3/PruSINU3 (master)
$
```

- Con el comando “git status”, vemos un ejemplo de un archivo nuevo no rastreado, que ha sido añadido al área de preparación mediante el comando “git add ruta_archivo”, y ahora está preparado y pendiente de consolidar en el repositorio.

```
MINGW64:/c/Florida/Docencia/22-23/1Semi-DAM-SIN/Contenidos/U3/PruSINU3 (master)
$ git add nuevo.txt

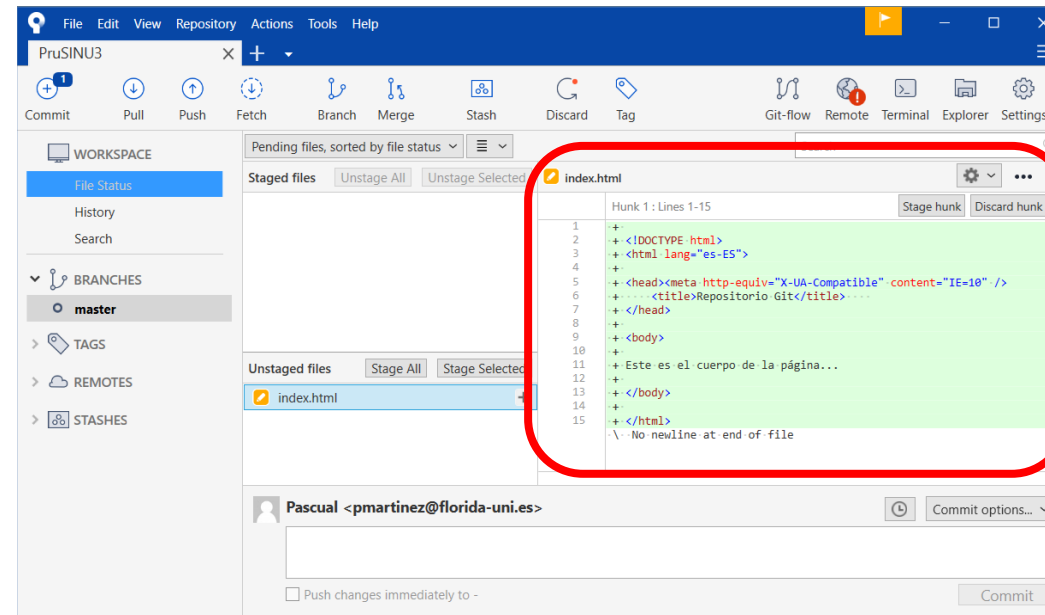
pmart@LAPTOP-LDPE3MBP MINGW64 /c/Florida/Docencia/22-23/1Semi-DAM-SIN/Contenidos/U3/PruSINU3 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   nuevo.txt

pmart@LAPTOP-LDPE3MBP MINGW64 /c/Florida/Docencia/22-23/1Semi-DAM-SIN/Contenidos/U3/PruSINU3 (master)
$
```

Git. SourceTree. Detalle de los cambios

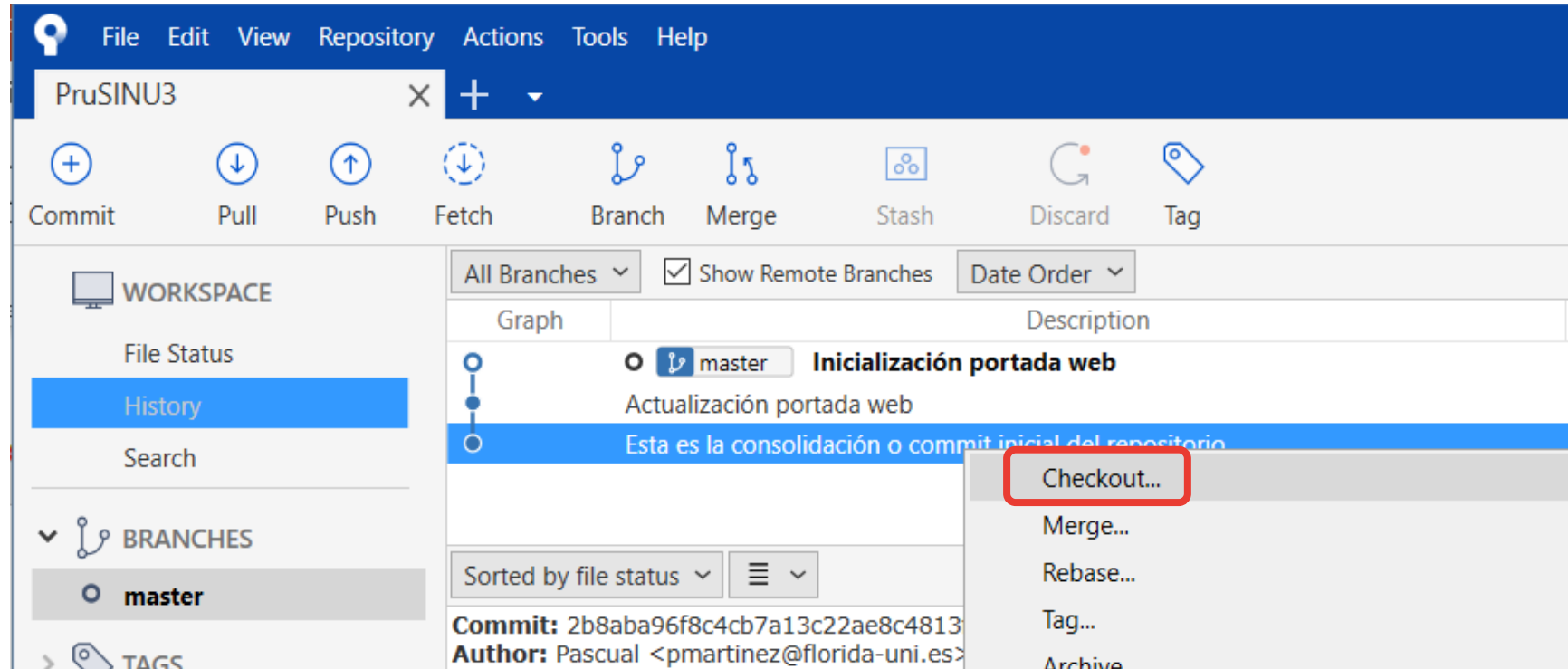
- Cuando realicemos **modificaciones** a los archivos **de nuestra carpeta del proyecto (directorio de trabajo)** en el sistema de archivos local, que **ahora es un repositorio Git**, SourceTree detectará estas modificaciones de forma automática. Podremos ver el detalle de los cambios realizados en cada fichero.



Git. Checkout

- Cada vez que realizamos una **consolidación** o commit, **Git crea un registro o instantánea**, que nos permitirá volver a ese estado del proyecto posteriormente.
- La acción de **restaurar un proyecto a un momento distinto del actual**, se denomina **checkout**.
- En **SorceTree**, si nos situamos en una consolidación o commit del historial (History) y pulsamos el botón derecho del ratón, se nos abre un menú desplegable, donde encontraremos la opción “**Checkout**”. Al pulsarla volveremos al estado del proyecto, en el momento de haber realizado esa consolidación en el repositorio.

Git. Checkout



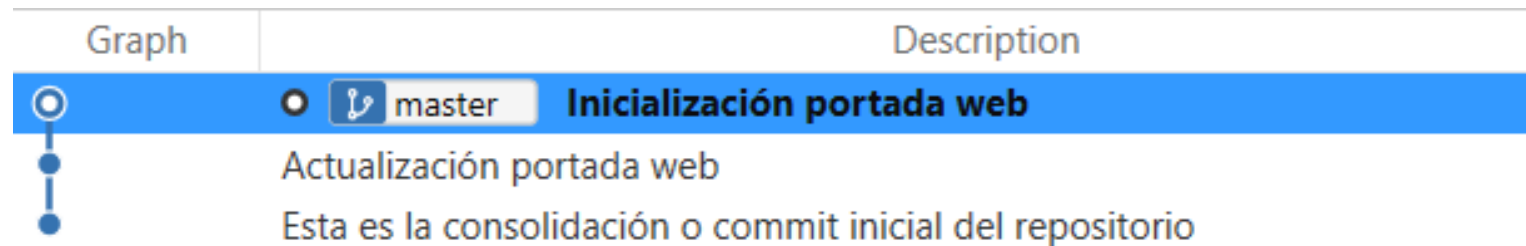
Git. Checkout

- **Vamos a ver un ejemplo:**
 - Supongamos que vamos a desarrollar un portal web y hemos generado un **nuevo repositorio** desde una carpeta local, con una serie de archivos básicos y a través de una **consolidación inicial**.



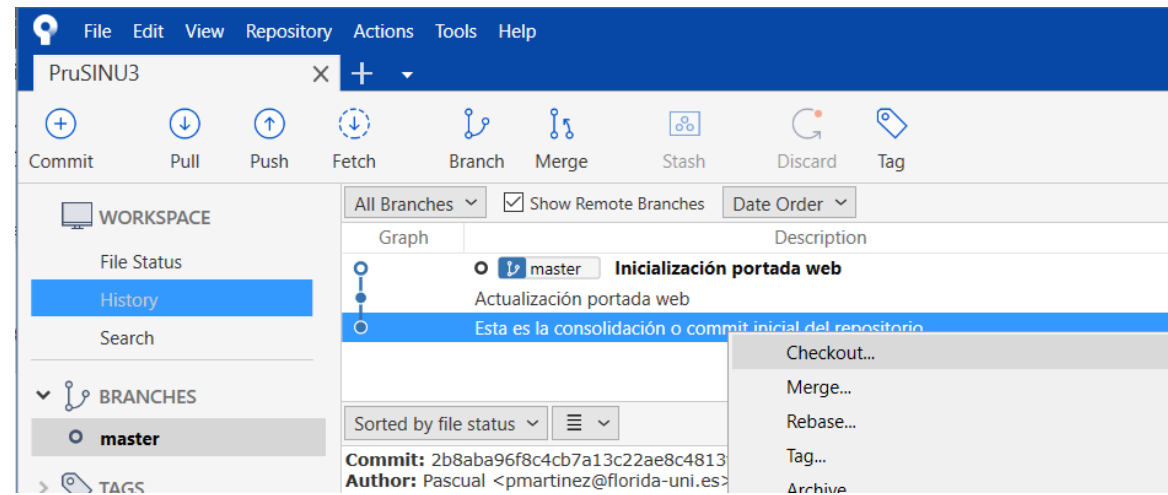
Git. Checkout

- **Vamos a ver un ejemplo:**
 - A continuación, realizamos **dos consolidaciones posteriores**, por ejemplo, referentes a cambios en la portada web y las describimos como:
 - Primero: **Actualización portada web**. Donde aportamos unas modificaciones de diseño.
 - Después: **Inicialización portada web**. Donde aportamos ciertos contenidos nuevos a la portada.



Git. Checkout

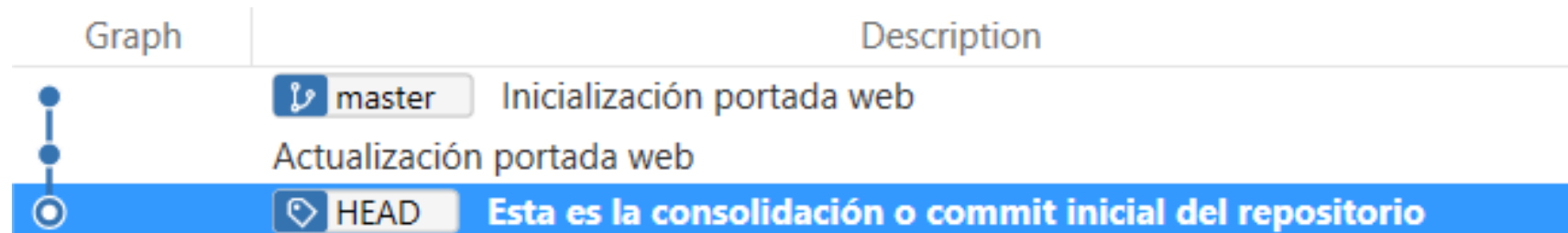
- **Vamos a ver un ejemplo:**
 - A continuación, **se decide que los cambios aplicados a la portada no deberían haberse realizado y que el proyecto debe volver al estado previo a realizar esos cambios.**
 - Como disponemos de una acción que permite restaurar un proyecto a una consolidación distinta de la actual (**checkout**), la podemos usar.



Git. Checkout

- **Vamos a ver un ejemplo:**

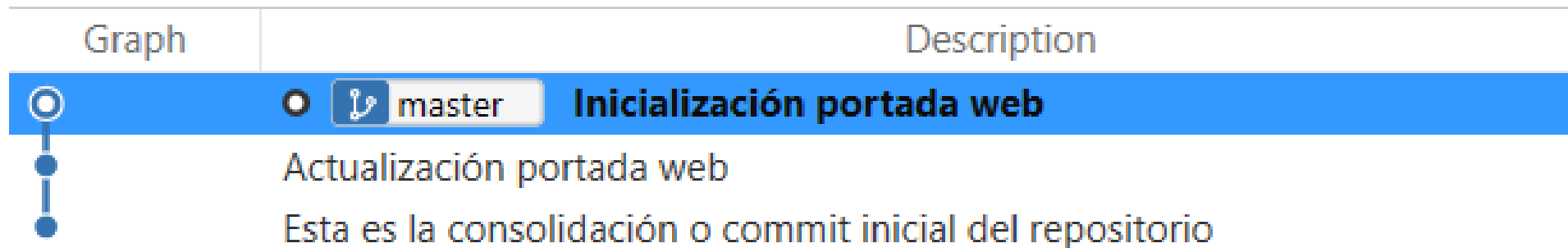
- Una vez realizado “checkout” a la primera consolidación, los archivos de nuestro proyecto estarán en el mismo estado que cuando hicimos la consolidación inicial.
- Aparece la etiqueta **HEAD**, que nos indica que el **repositorio** está **posicionado en un estado o consolidación diferente al más reciente**, es decir anterior.
- O lo que es lo mismo, existen consolidaciones del repositorio más recientes, aunque no están activas en este momento.



Git. Checkout

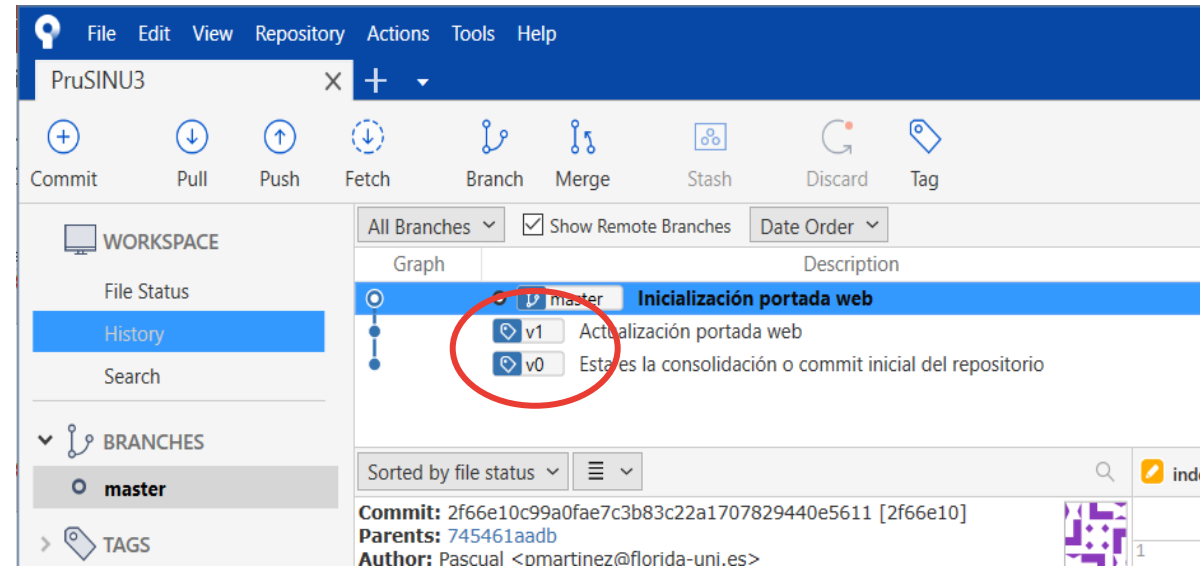
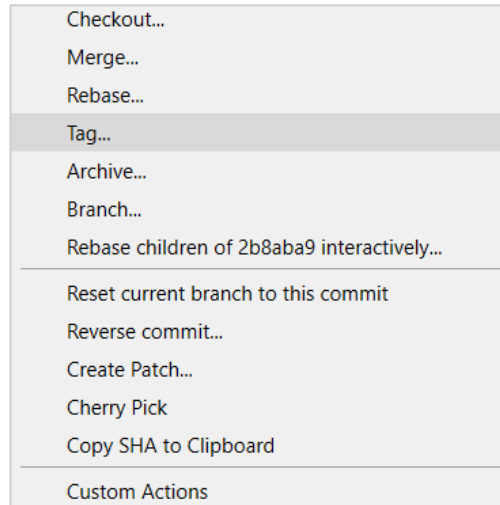
- **Vamos a ver un ejemplo:**

- Si en este momento, se decide que se deben recuperar los cambios que se habían realizado a la portada web, volveríamos a hacer “checkout” en la consolidación con la descripción “Iniciación portada web”.
- Al hacerlo, el repositorio se posiciona de nuevo en la consolidación más reciente y desaparecería la etiqueta **HEAD**.



Git. Tags

- Disponemos de **etiquetas** para poder **identificar las consolidaciones**, se denominan **Tags**.
- Podemos asociar una etiqueta (por ejemplo, v0, v1, ...) a aquellas consolidaciones (commits) que nos resulte útil identificar. Mediante botón derecho, opción **“Tag”**.



Git. Gitignore

- Es posible indicarle a Git que podemos necesitar **disponer de archivos** en nuestra carpeta de proyecto, **que no queremos que se integren** en el repositorio.
- Por ejemplo, porque son ficheros temporales de trabajo o de contraseñas o privados y no para consolidar en el repositorio.
- Esto se gestiona creando un fichero llamado **.gitignore** e indicando en él qué ficheros debe ignorar Git.

 **gitignore**

