

Titulo

Manual del Usuario

Breve Descripción

Manual de usuario para la herramienta GIT dedicada al control de versiones en los distintos proyectos.

Propósito

La creación de este manual se realiza con el propósito de dar a conocer a detalle el uso de la herramienta y las funciones que pueden utilizarse enlazadas con los host de versiones.

Elaborado por

SAXA

Ultima revisión

11/03/2020

Referencias

Versión

1.0

Válido

De: *Diciembre/2020*

A: *Diciembre/2021*

Sustituye al Documento

Estado

☐ ☐ Draft

☐ ☒ Definitive

Lista de Distribución

SC Manager	RGA
Scrum Master	COMS
Software Project Manager	COMS
Technical Leader	MORR
SCM Group	BEGI, HEOX
SQA Group	RURL, TOBF
Software Engineering Group	MORR, MALZ
Test Group	ANBM, SARE, RITL, LORO
Data Analyst	MROM

Autorización

Elaborado por

SAXA

Revisado por

Revisado por / Autorizado por

1. INTRODUCCIÓN	3
1.1 GIT	3
1.2 ÁREAS DE GIT	3
2. COMANDOS BÁSICOS	4
3. USO DE GIT	5
3.1 AGREGAR ARCHIVOS AL REPOSITORIO	5
3.1.1 Añadir todos los archivos	5
3.2 REALIZAR COMMIT	5
3.3 DESCARTAR CAMBIOS	6
3.4 VER DIFERENCIAS ENTRE ARCHIVOS	7
3.5 MOSTRAR VERSIONES	7
3.6 IGNORAR CARPETAS Y ARCHIVOS	8
3.7 CREAR PROYECTOS ALTERNATIVOS	11
4. COMPARTIR PROYECTOS	15
4.2 GitHub	19
5. INSERTAR INFORMACIÓN DE AYUDA AL COLABORADOR O USUARIO	24
6. RECUPERAR PROYECTOS	26
7. AGREGAR COLABORADORES	27
7.1 BAJAR CAMBIOS DE LOS COLABORADORES	28

1. INTRODUCCIÓN

En el presente documento se dan a conocer las instrucciones para el uso adecuado de la herramienta Git, la cual es requerida para llevar a cabo el control de versiones de los proyectos en Praxis.

El manual presenta una breve descripción acerca de Git, posteriormente se explican las áreas que lo componen, así como una serie de comandos más comunes a utilizar, después de ello se desglosan las funciones y su uso correspondiente.

Este manual está basado en la versión 2.25.1.

1.1 GIT

Git es un sistema distribuido de control de versiones para rastrear cambios en el código fuente durante el desarrollo de software. Está diseñado para coordinar el trabajo entre programadores, pero se puede usar para rastrear cambios en cualquier conjunto de archivos. Sus objetivos incluyen velocidad, integridad de datos y soporte para flujos de trabajo no lineales distribuidos.

1.2 ÁREAS DE GIT

- ✚ Directorio de Trabajo: Es donde se trabajan todos los archivos.
- ✚ Área de prueba: Se preparan los archivos siendo versionados.
- ✚ Repositorio: Son los cambios guardados.

2. COMANDOS BÁSICOS

git init: Indica que comenzaremos a utilizar GIT.

git add <file>: Pasa los archivos del Working Directory al Staging área.

git status: Ver en qué área se encuentran los archivos.

git commit: Pasa los archivos de Staging área al Repositorio.

git push: Sube los archivos a un repositorio remoto.

git pull: Trae los cambios que han hecho los colaboradores.

git clone: Hace una copia desde el servidor central donde está el código a nuestras computadoras para comenzar a trabajar.

pwd: Indica en qué directorio está ubicado.

ls: Lista los archivos que se encuentran dentro del directorio.

3. USO DE GIT

- Por default, al abrir el bash o la consola de comandos, le coloca en la carpeta de usuario con su nombre determinado. A partir de ahí, puede entrar a cualquier ruta con el comando **cd**

Ejemplo:

```
cd destkop/MiPagina
```

- Una vez posicionado en la dirección de la carpeta en la que estará trabajando puede escribir el comando **git init**, de esa manera, indica que esa carpeta será versionada y/o subida a un repositorio.

3.1 AGREGAR ARCHIVOS AL REPOSITORIO

- Para que GIT tome en cuenta todos los archivos de la carpeta agregue **git add <file>**

Ejemplo:

```
git add index.html
```

```
git add app.js
```

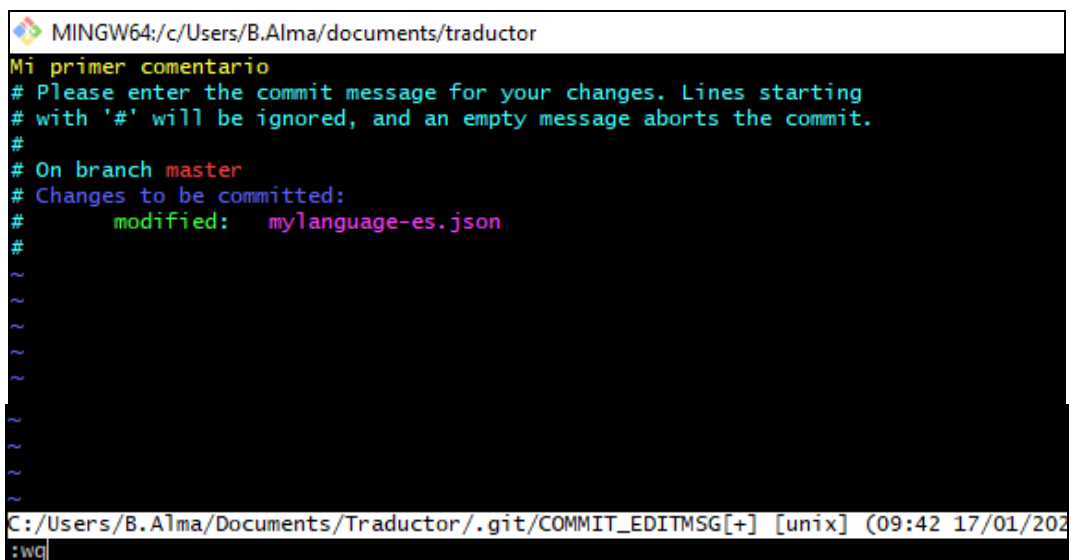
3.1.1 Añadir todos los archivos

Para evitar que añada archivos uno a uno, puede colocar el comando **git add .** para agregar todos los archivos del directorio del proyecto al Área de prueba.

3.2 REALIZAR COMMIT

- Ya que todos los archivos de la carpeta fueron agregados puede trabajar sobre ellos y al guardarlos debe escribir el comando **git commit**.

- Le abrirá una nueva ventana en donde pedirá un comentario acerca de la versión que estamos subiendo, para poder comenzar a escribir debe presionar primero la tecla “i”.
- Cuando lo haya escrito presione la tecla **Esc**, seguida de **dos puntos** y las letras **wq** (Write y Quit) para que los cambios sean guardados y vuelva a la línea de comandos del bash.



```
MINGW64:/c/Users/B.Alma/documents/traductor
Mi primer comentario
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#   modified:   mylanguage-es.json
#
~
~
~
~
~
~
~
~
~
~
C:/Users/B.Alma/Documents/Traductor/.git/COMMIT_EDITMSG[+] [unix] (09:42 17/01/202
:wq|
```

3.3 DESCARTAR CAMBIOS

- Si ha realizado un cambio en alguno de sus archivos y desea deshacerlos antes de agregar un archivo y hacer un commit, coloque el comando:

git checkout -- <nombre del archivo>

3.4 VER DIFERENCIAS ENTRE ARCHIVOS

Para mostrar las diferencias entre el último cambio al archivo y el anterior antes de agregar un archivo y hacer un commit, puede escribir el comando:

```
git diff <file>
```

Ejemplo

```
git diff index.html
```

3.5 MOSTRAR VERSIONES

- Para ver las distintas versiones del directorio puede colocar el siguiente comando, como resultado nos mostrará una clave única por cada versión para identificarlas con mayor facilidad:

```
git log
```

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git log
commit bc5f5a3566aaebb5711e6f96293de146799d8c5b (HEAD -> master)
Author: Alma <tu_email_aquí@example.com>
Date: Thu Jan 16 14:11:12 2020 +0100

    pruebita

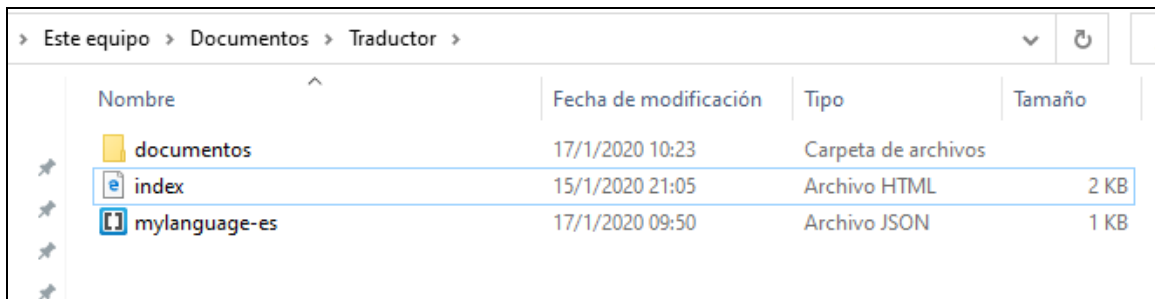
commit cf6891665b7e098405a96d07715729b05a88a7c0
Author: Alma <tu_email_aquí@example.com>
Date: Wed Jan 15 17:23:28 2020 +0100

    Mi primer commit
```

3.6 IGNORAR CARPETAS Y ARCHIVOS

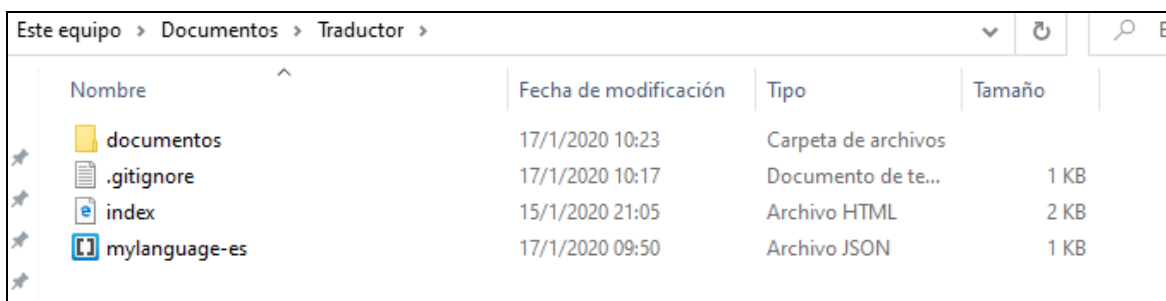
Puede darse el caso que usted tenga distintas carpetas dentro del directorio donde está trabajando GIT, y algunas de ellas quisiera descartarlas para no afectarlas ni realizar ningún cambio en ellas.

Por ejemplo:



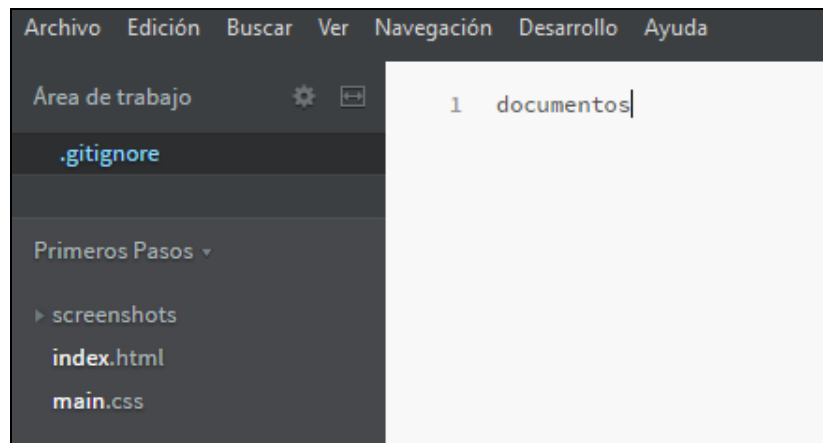
Este equipo > Documentos > Traductor >				
Nombre	Fecha de modificación	Tipo	Tamaño	
documentos	17/1/2020 10:23	Carpeta de archivos		
index	15/1/2020 21:05	Archivo HTML	2 KB	
mylanguage-es	17/1/2020 09:50	Archivo JSON	1 KB	

- En este caso, para ignorar la carpeta “documentos” debe crear un archivo en un editor de código (funciona con cualquier editor de código), aquí se usa el software Brackets. Escriba de nombre al nuevo archivo **“.gitignore”** y guárdelo en la raíz principal de su carpeta que está trabajando con GIT, en este caso será en la carpeta “Traductor”.



Este equipo > Documentos > Traductor >				
Nombre	Fecha de modificación	Tipo	Tamaño	
documentos	17/1/2020 10:23	Carpeta de archivos		
.gitignore	17/1/2020 10:17	Documento de te...	1 KB	
index	15/1/2020 21:05	Archivo HTML	2 KB	
mylanguage-es	17/1/2020 09:50	Archivo JSON	1 KB	

- Dentro del archivo `.gitignore` creado, solo escriba el nombre de la carpeta que desea descartar para cambios y guárdelo. En este ejemplo se ignora la carpeta “documentos” que tenemos dentro del directorio “Traductor”.



- En la consola de GIT escriba **git status** y le arrojará el nombre del archivo que acaba de crear en color rojo, en lugar del directorio ignorado.
- A continuación se muestra el resultado en consola antes y después de descartar la carpeta:

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    documentos/

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

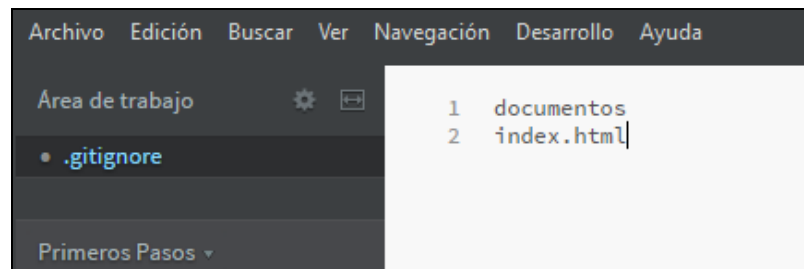
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
```

- Ahora lo agregamos como un archivo normal a GIT, de nuevo con el comando
git add <file>

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git add .gitignore

B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore
```

- Este procedimiento funciona con carpetas y archivos, si desea ignorar otra carpeta u otro archivo solo escríbalo dentro de **.gitignore** y guarde.



- De esta manera, puede manejar el archivo **.gitignore** como cualquier otro y por lo tanto es necesario también añadirlo a GIT y hacer commit para que se guarde y quede versionado.

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git add .gitignore

B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git commit
[master 3b62175] Probando gitignorProbando gitignore
1 file changed, 1 insertion(+)
create mode 100644 .gitignore
```

*****Nota:** Al hacer commit en cualquier archivo, puede evitar la ventana de comentario escribiendo el siguiente comando:

git commit -m "agregando un git ignore"

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git commit -m "Se agrega un gitignore"
```

3.7 CREAR PROYECTOS ALTERNATIVOS

- Si requiere hacer una copia del proyecto completo para respaldarlo o trabajarlo de forma diferente sin perder el original, con el comando **git branch <nombre del respaldo>** crea automáticamente una copia de su proyecto o carpeta principal.

Ejemplo:

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git branch login
```

- Para visualizar las copias de su proyecto creadas, inserte la línea:

git branch

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git branch
  login
* master
```

- En donde master representa el proyecto original y el símbolo de asterisco le indica en cuál de ellos se encuentra ubicado.
- Para cambiar de ubicación entre las copias o el proyecto original debe escribir lo siguiente en consola:

git checkout <nombre de la copia en la que desea trabajar>

Ejemplo:

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git checkout login
Switched to branch 'login'
```

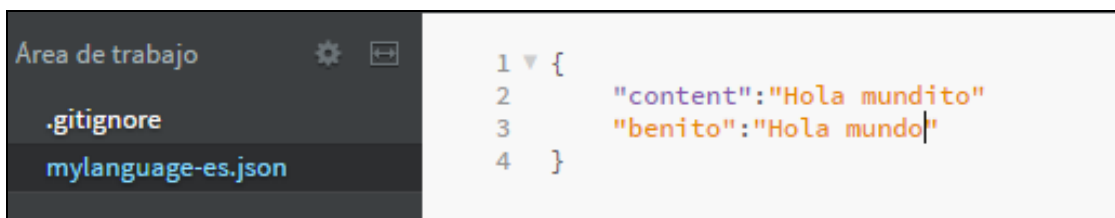
- Así, al ejecutar **git branch** notará que el símbolo de asterisco aparece señalando el proyecto que le haya indicado.

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (login)
$ git branch
* login
  master
```

- Después de este paso, puede comenzar a editar sobre la misma carpeta, sin que se modifique el contenido original.

Ejemplo:

- Estando dentro de la copia del proyecto **Login** se comienza a editar el mismo archivo que se ha usado “mylanguage-es.json” y se agrega una nueva línea.



```
1 {
2   "content": "Hola mundito"
3   "benito": "Hola mundo"
4 }
```

Cuando escriba **git status** en la consola, aparecerá que se ha modificado el archivo (No olvide que se está trabajando sobre la copia creada).

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (login)
$ git status
On branch login
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:   mylanguage-es.json

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

- Como aparece en rojo, se debe añadir el archivo y hacer el commit, como se ha mostrado en este documento.

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (login)
$ git add mylanguage-es.json

B.Alma@201701775-FRT MINGW64 ~/documents/traductor (login)
$ git commit -m "Modificando la copia con login"
[login c14299e] Modificando la copia con login
1 file changed, 1 insertion(+)
```

- Pareciera que se ha modificado el proyecto original, sin embargo al colocar **git log** en la consola se muestra que el último commit fue realizado en la copia **Login** y no en el original. Ya que la última versión finaliza con el texto (**HEAD - > login**) y no en **master**.

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (login)
$ git log
commit c14299eeb34ccf8e8a5a40aaa1b523c13d6a1dab (HEAD -> login)
Author: Alma <tu_email_aquí@example.com>
Date:   Fri Jan 17 13:20:30 2020 +0100

    Modificando la copia con login

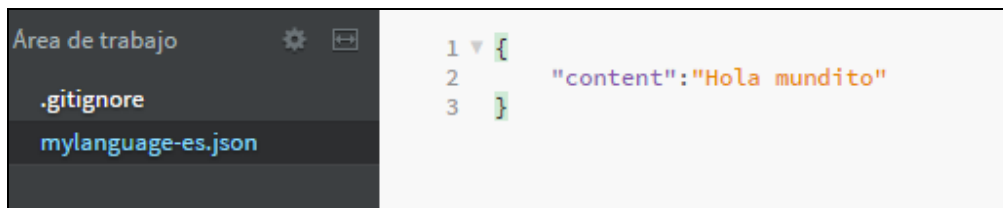
commit 3b62175dd8013ba4e2630abb0c6b18843db55df9 (master)
Author: Alma <tu_email_aquí@example.com>
Date:   Fri Jan 17 11:10:29 2020 +0100

    Probando gitignorProbando gitignore

commit 2c6c2c019aa704bb4a332edc039cba0d63704478
Author: Alma <tu_email_aquí@example.com>
Date:   Fri Jan 17 09:42:36 2020 +0100

    Mi primer comentario
```

- Si se vuelve al proyecto original escribiendo **git checkout master**, se puede notar que el archivo que se había modificado, regresó automáticamente a su estado original, ya que el último commit nunca afectó a la carpeta original.



4. COMPARTIR PROYECTOS

Para que varios usuarios puedan modificar o trabajar sobre el repositorio, es necesario compartirlo a través de servicios de hosting, para que cada usuario guarde la versión del proyecto original, realice cambios y los suba a través de otra instancia de Git.

En este punto, es de suma importancia saber qué procedimientos tiene como estándar el equipo de trabajo donde se encuentre, por ejemplo, **será necesario que cada usuario tenga una instancia de GIT instalada en su computadora en caso de no contar con ninguna extensión de GitLab o GitHub en modo escritorio**, esto con motivo de trabajar de manera unificada y a prueba de errores en cuanto al versionamiento de los proyectos a través de ramificaciones (branch).

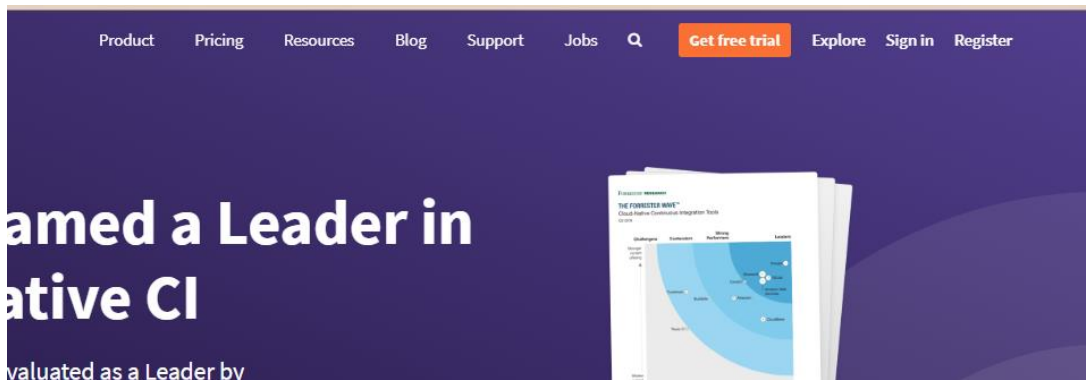
Por otra parte, **sería posible utilizar una sola instalación de GIT en un servidor y crear las distintas ramificaciones en el mismo, de esta manera, cada usuario accedería al mismo servidor pero solo a la ramificación que le corresponde**, sin embargo, debe considerar que eso podría traer problemas en cuanto a saturación en el servidor, además de que las personas que trabajen las ramificaciones deben ser expertos manejando la herramienta para no obtener errores o modificar la ramificación de alguien más.

4.1 GitLab

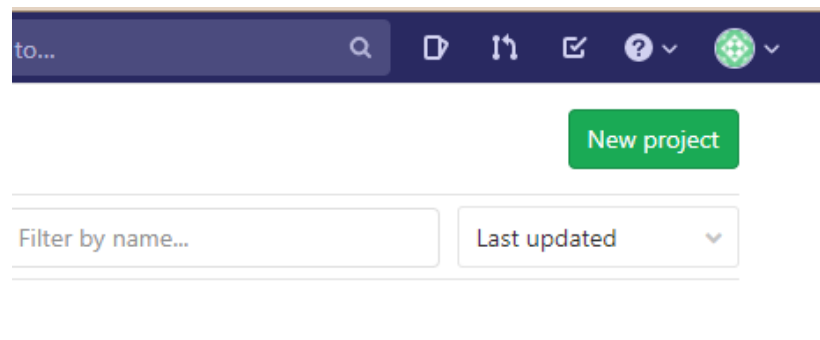
GitLab es una plataforma de hosting enfocada especialmente a nivel empresarial para desarrollar y compartir proyectos a partir del controlador de versiones GIT.

- Para subir y compartir sus proyectos con otros colaboradores, primero debe conocer de qué manera está implementado GitLab, si ha sido instalado localmente en un servidor debe tener acceso a la dirección web que se le haya asignado. En caso de no haber sido instalado y estar disponible por medio de la nube basta solo con ingresar a la página <https://gitlab.com>

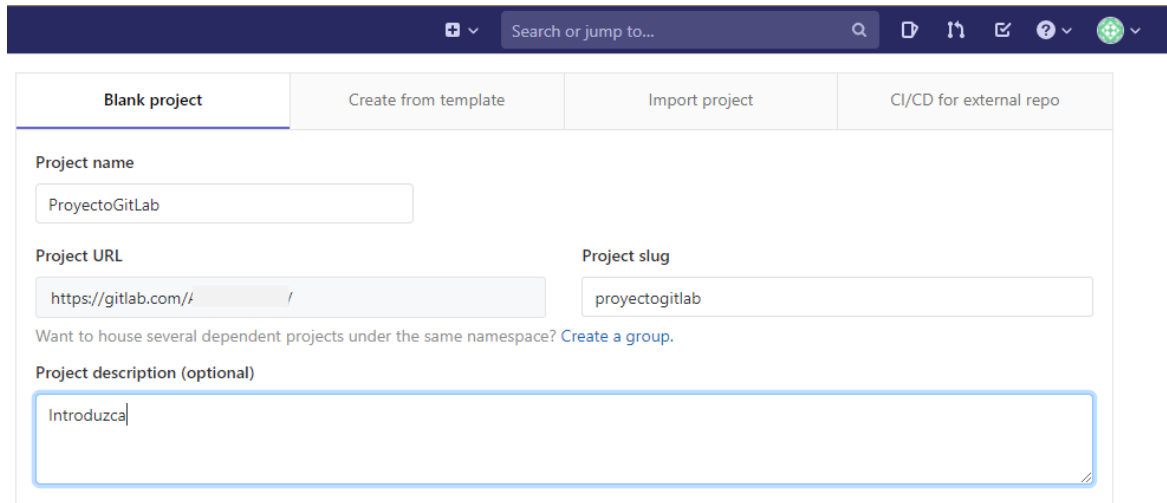
- Si antes ya se ha registrado en esta página, solo haga clic en la esquina superior derecha en el botón “Sign in” para acceder a su cuenta.
- En caso de no estar registrado, haga clic en el botón “Register” para crear un nuevo perfil.



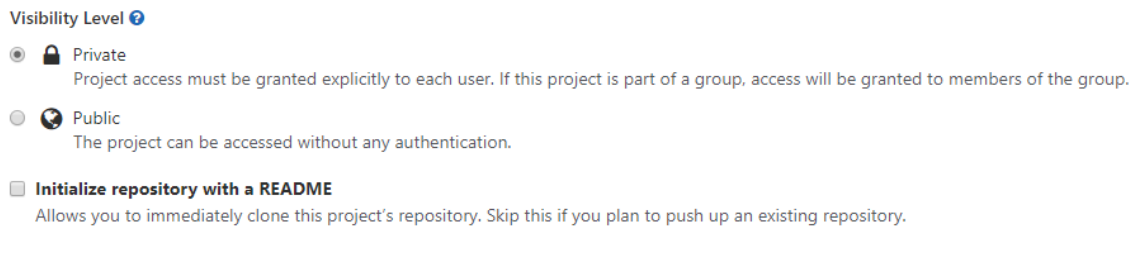
- Una vez que haya ingresado con su perfil a GitLab, en la pantalla principal diríjase al botón verde de “Nuevo Proyecto” y haga clic.



- Abrirá una ventana con una serie de campos a llenar, en la cual se debe colocar el nombre del proyecto y una descripción de este.



- En la parte de abajo puede elegir la modalidad en la que desea su proyecto, público o privado.



- Finalmente haga clic en el botón inferior que dice “Crear repositorio”.
- Mostrará una serie de líneas de comandos que deben insertarse en la consola de GIT para llevar a cabo el repositorio de su proyecto, pero solo debe enfocarse en algunos de ellos.
- El recuadro para crear un nuevo repositorio es en caso de haber sido invitado a modificar el código y debe descargar el proyecto por primera vez, si es así, coloque las líneas de comando de este apartado una por una.

Create a new repository

```
git clone https://gitlab.com/AlmaSauceda/proyectgitlab.git
cd proyectgitlab
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

- El siguiente recuadro es para compartir un folder o proyecto existente dentro del equipo de cómputo, al igual que el anterior, se coloca línea por línea en la ventana de comandos de GIT.

Push an existing folder

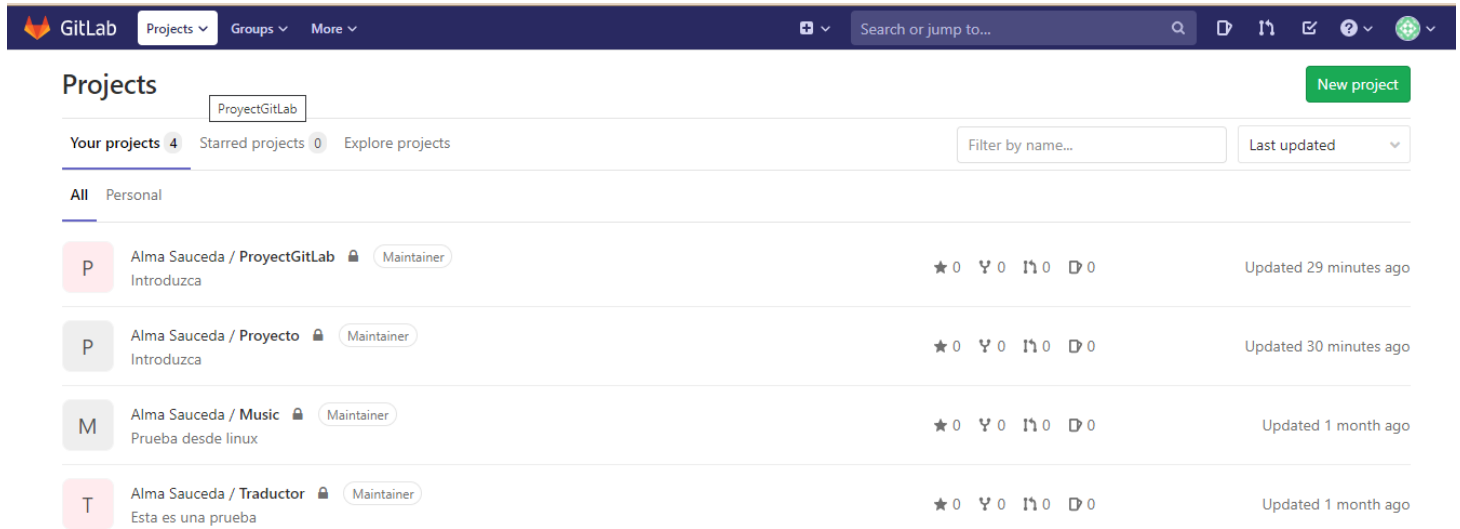
```
cd existing_folder
git init
git remote add origin https://gitlab.com/AlmaSauceda/proyectgitlab.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

- Existe otro recuadro en la parte inferior, este es para subir un repositorio que ya tengamos creado en GIT a la plataforma de hosting.

Push an existing Git repository

```
cd existing_repo
git remote rename origin old-origin
git remote add origin https://gitlab.com/AlmaSauceda/proyectgitlab.git
git push -u origin --all
git push -u origin --tags
```

- Luego de eso, su repositorio habrá sido agregado y vinculado con éxito y al refrescar la pantalla de inicio de GitLab notará que se ha añadido a la lista de sus proyectos.



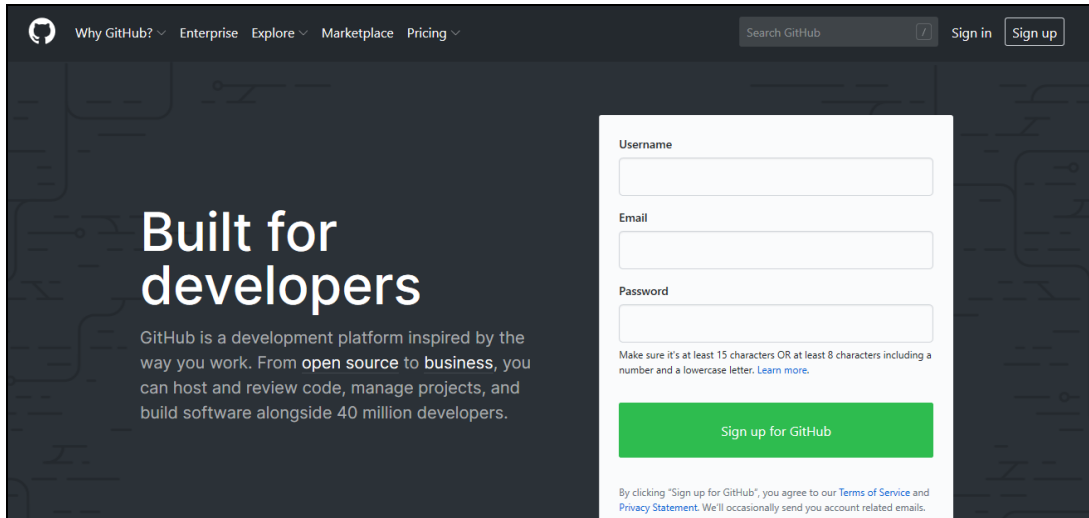
The screenshot shows the GitLab web interface. At the top is a dark blue navigation bar with the GitLab logo, tabs for 'Projects', 'Groups', and 'More', a search bar, and various utility icons. Below the navigation bar, the 'Projects' section is active, displaying a list of projects under the 'Your projects' tab. The list includes four projects: 'ProjectGitLab', 'Proyecto', 'Music', and 'Traductor'. Each project entry shows its name, a 'Maintainer' badge, and statistics for stars, forks, merges, and deployments. The 'ProjectGitLab' project is highlighted with a pink background.

*****NOTA:** GitLab cuenta con distintos protocolos para la conexión servidor-cliente, puede ser sencilla por medio de HTTPS o por medio de SSH, el cual es un protocolo más seguro sobre todo a nivel empresarial.

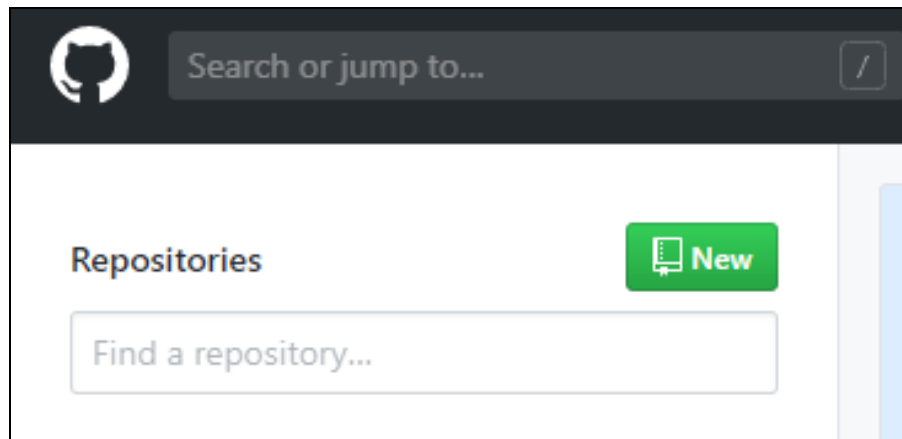
4.2 GitHub

Para este paso, se requiere de una herramienta muy relacionada a GIT llamada GitHub, la cual es una plataforma para el hosting de los proyectos entre una comunidad de personas que desarrollan y comparten, usando GIT.

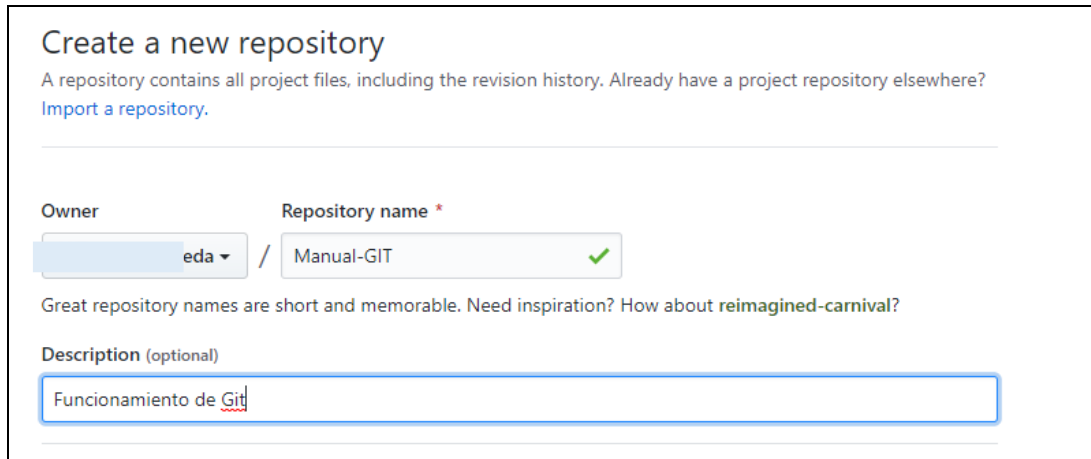
- Para subir y compartir sus proyectos con otros colaboradores, tiene que ingresar en su navegador web a la página <https://github.com>
- Si ya se ha registrado en dicha página anteriormente solo haga clic en la esquina superior derecha en el botón "Sign in" para acceder a su cuenta.
- En caso de no estar registrado, cree un nuevo perfil llenando los datos que le piden en la página principal.



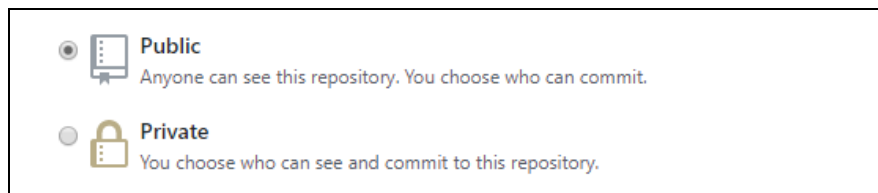
- Una vez que haya ingresado con su perfil a GitHub, diríjase al apartado de “Repositorios” y de clic en “Nuevo”.



- En el nombre del repositorio escriba el que usted prefiera, solo cuide el no repetir el nombre de algún repositorio ya existente (En caso de haber contado con perfil de GitHub anteriormente).
- Debajo puede colocar una descripción de su proyecto.



- Después, elija si quiere que sus archivos sean públicos o privados, es decir, si su proyecto es público cualquier persona podrá ver el contenido de la carpeta, sin hacer commits. Mientras que en privado, puede determinar quiénes pueden visualizar el contenido y hacer commits. No obstante, esta última opción solo aplica si tiene una suscripción especial, y por lo tanto lleva un costo monetario de por medio.



- Finalmente haga clic en el botón inferior que dice “Crear repositorio”.
- Mostrará una serie de líneas de comandos que deben insertarse en la consola de GIT para llevar a cabo el repositorio de su proyecto, pero solo debe enfocarse en algunos de ellos.
 - El primero es:

```
git remote add origin https://github.com/AlmaSauceda/Manual-GIT.git
```

Copie y péguelo en consola para indicar al proyecto en dónde se va almacenar.

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git remote add origin https://github.com/AlmaSauceda/Manual-GIT.git
```

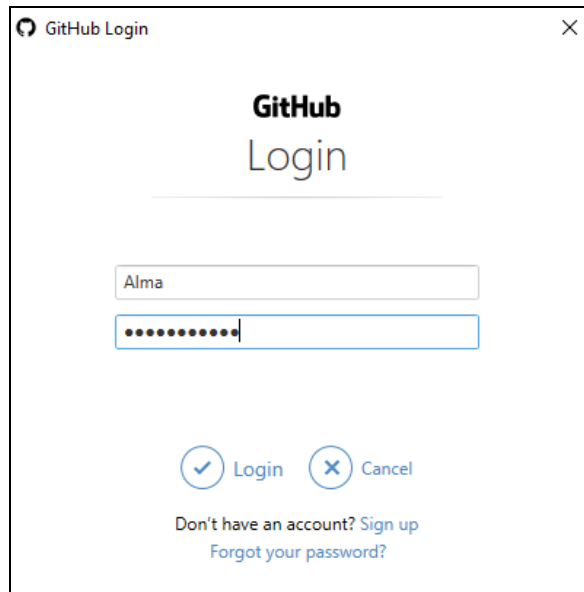
- El segundo es:

```
git push -u origin master
```

Copie y péguelo en consola para lanzar su proyecto a GitHub.

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git push -u origin master
```

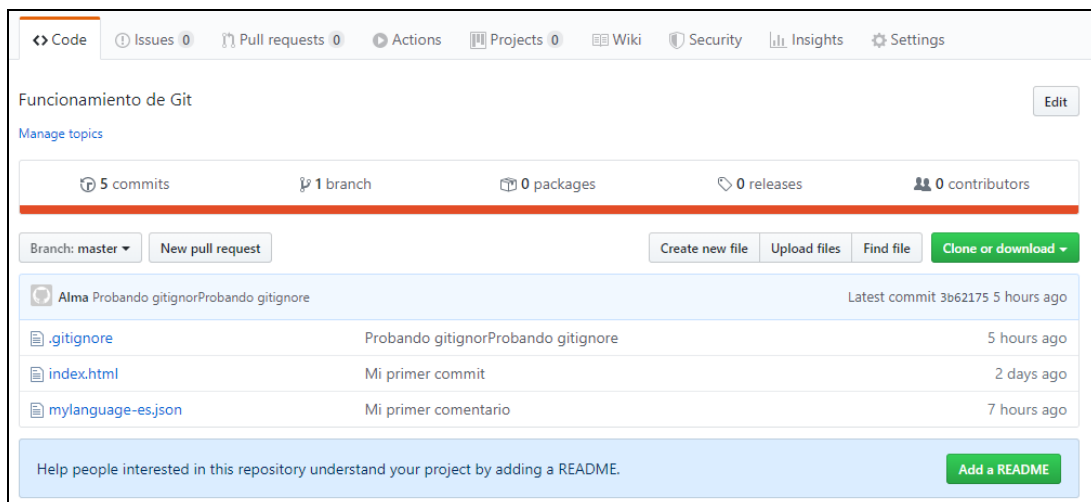
- Le abrirá una pequeña ventana de GitHub, donde le pedirá el usuario y contraseña de su perfil de GitHub.



- Presione “Login”.
- De inmediato en su consola de GIT comenzará a subir todos los archivos de su proyecto a GitHub.

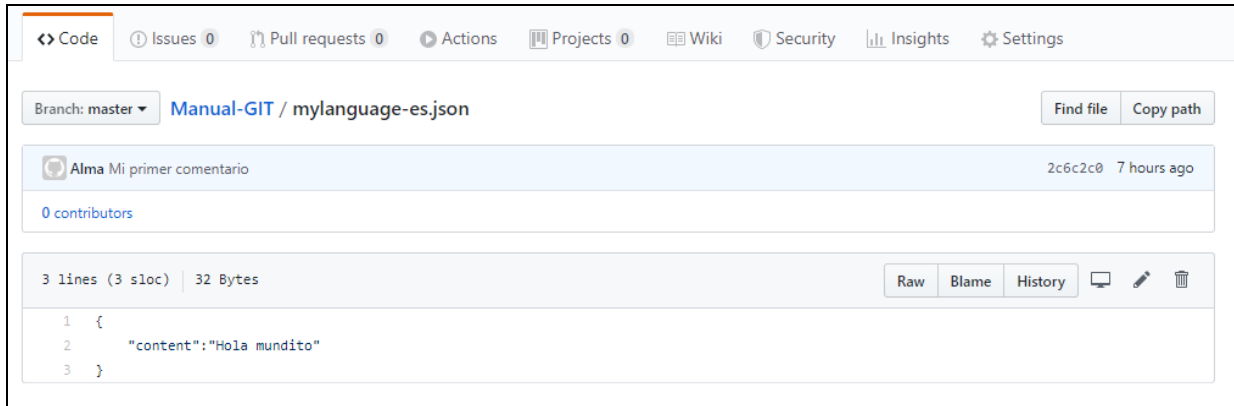
```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git push -u origin master
Logon failed, use ctrl+c to cancel basic credential prompt.
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (12/12), 1.61 KiB | 548.00 KiB/s, done.
Total 12 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/AlmaSauceda/Manual-GIT.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

- Luego, refresque la página de su repositorio en GitHub, y aparecerá todo el proyecto y los cambios que ha realizado.



The screenshot shows the GitHub interface for a repository named "Funcionamiento de Git". The top navigation bar includes links for Code, Issues (0), Pull requests (0), Actions, Projects (0), Wiki, Security, Insights, and Settings. Below the repository name, there are statistics: 5 commits, 1 branch, 0 packages, 0 releases, and 0 contributors. A section for the "master" branch shows a list of files: ".gitignore" (5 hours ago), "index.html" (2 days ago), and "mylanguage-es.json" (7 hours ago). At the bottom, there is a prompt to "Add a README" to help people understand the project.

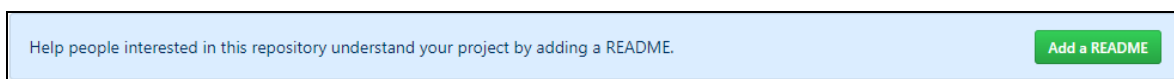
- Al hacer clic en cada archivo podrá visualizar el código o el contenido en cada uno de ellos.



5. INSERTAR INFORMACIÓN DE AYUDA AL COLABORADOR O USUARIO

En varias ocasiones es importante dar a entender el proyecto detalladamente para que los colaboradores sepan el contexto y lo trabajen de manera correcta, y los usuarios logren aterrizar toda la información que se les esté proporcionando. Dentro de esta información se debe de encontrar el ID, la fecha y una referencia al requerimiento que se está trabajando o descripción breve en caso de ser la corrección de un código.

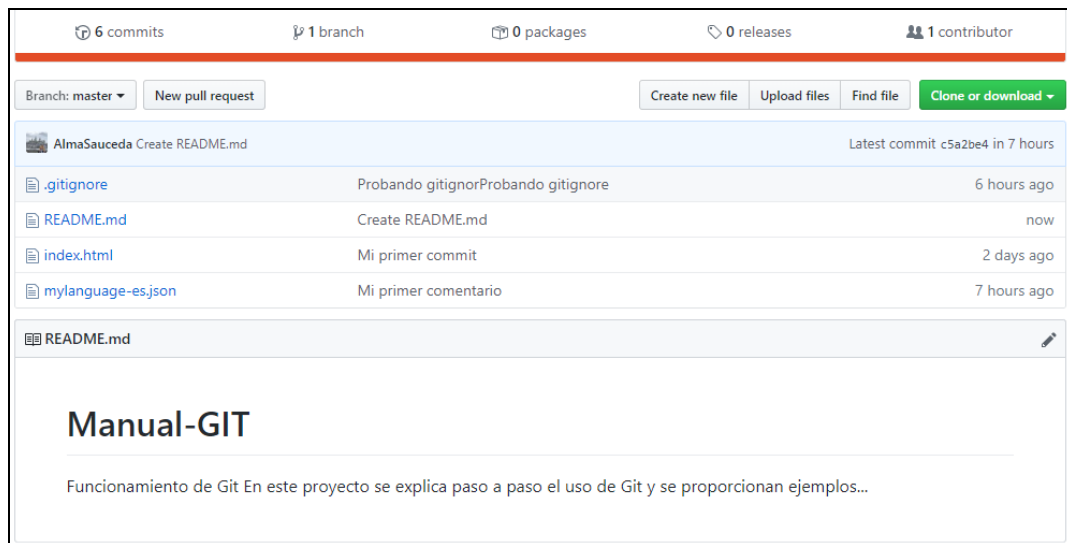
- En la parte inferior donde se encuentra su proyecto en GitHub hay una opción que dice “Agregar un README”.



- Al hacer clic en ese botón, puede agregar la información necesaria sobre su proyecto y archivos.




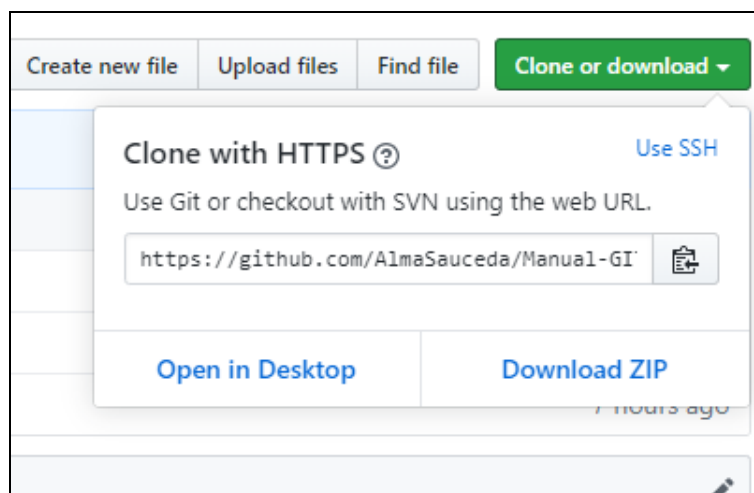
- Finalice el escrito dando clic en el botón inferior **“Confirmar nuevo archivo”**.
- Y debajo de los directorios aparecerá la información que haya escrito acerca de su proyecto, además, se agrega un archivo llamado README.md (Si es que conservó el nombre y no lo reemplazó), esto no afecta a su código, solo contiene la información que se le proporcione.



6. RECUPERAR PROYECTOS

Si por accidente ha eliminado su proyecto de GIT y de su computadora, podrá recuperarlo desde GitHub.

- En la página donde GitHub muestra su proyecto, aparece una opción en botón verde que es “Clonar o Descargar”, de clic ahí.
- Muestra una pequeña ventana, oprima el botón  para copiar la ruta de su carpeta compartida en GitHub.



- Abra la consola de GIT.
- Entre con el comando `cd` a la ruta donde quiere guardar su proyecto.
- Pegue la dirección después del comando **git clone**

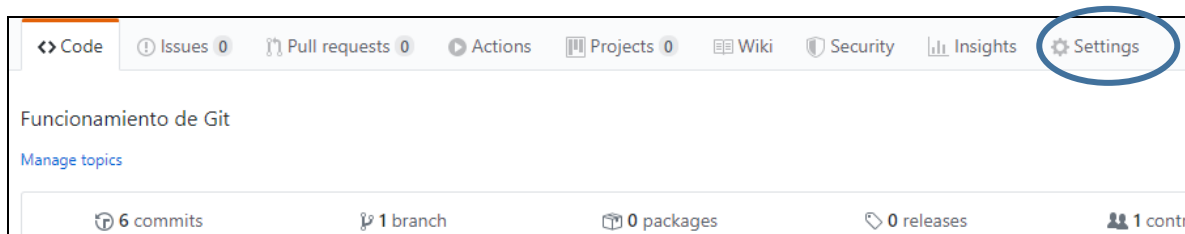
```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git clone https://github.com/AlmaSauceda/Manual-GIT.git
```

- Y tendrá de nuevo todos los archivos de la carpeta en su ordenador.

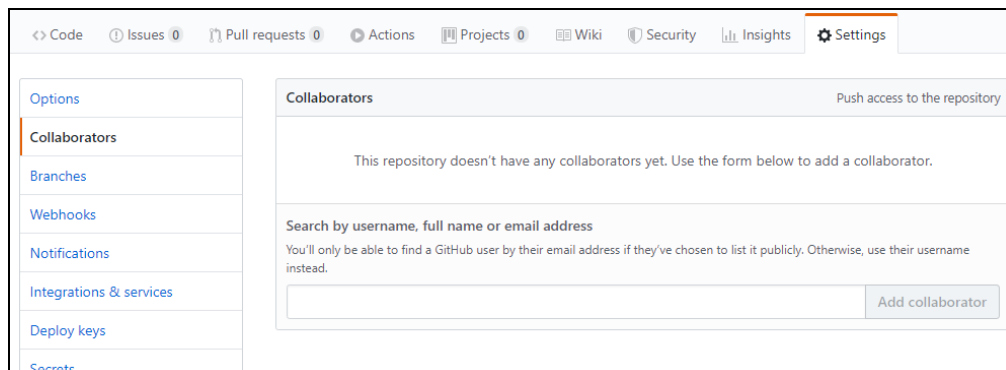
7. AGREGAR COLABORADORES

En un equipo de trabajo es común que sean varias personas que colaboren en el proyecto al mismo tiempo. Hasta ahora solo se ha mostrado cómo manejar los archivos individualmente, es momento de invitar a compañeros a trabajar en el proyecto.

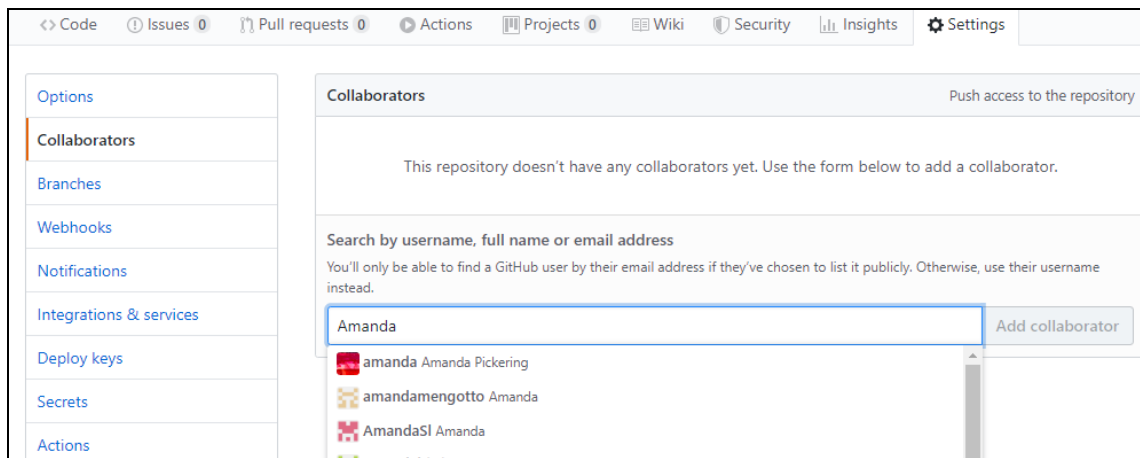
- En la página de GitHub donde aparece la carpeta y los archivos que subió, en la parte derecha superior aparece la opción de “Configuración”, deberá dar clic ahí.



- Por consiguiente, escoja el apartado de “Colaboradores” que se encuentra en el menú izquierdo de la pantalla.



- Y en la barra en blanco que se muestra, escriba el nombre de usuario de sus compañeros.



- Haga clic en “Añadir Colaborador”.
- Luego de eso, se le notificará a la persona que también esté registrada en GitHub por medio del correo electrónico que lo ha invitado a colaborar en un proyecto.
- Cuando la persona acepte la invitación tendrá acceso al repositorio y podrá modificar y versionar, de esta manera, podrán trabajar al mismo tiempo sin preocuparse de que el código se dañe o pierda cambios importantes.

7.1 BAJAR CAMBIOS DE LOS COLABORADORES

Cuando trabaje en equipo es importante guardar antes los cambios de los colaboradores y después de eso subir los cambios que haya hecho usted.

- Una vez que ya haya realizado todos los pasos anteriores, estará trabajando con el proyecto de forma satisfactoria, así que solo para guardar los cambios o commits de sus compañeros debe agregar el comando:



git pull origin master

- Guardará los cambios en su proyecto local y estará listo para que usted haga los commits necesarios.

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git pull origin master
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 827 bytes | 7.00 KiB/s, done.
From https://github.com/AlmaSauceda/Manual-GIT
* branch                master      -> FETCH_HEAD
   3b62175..c5a2be4      master      -> origin/master
Updating 3b62175..c5a2be4
Fast-forward
 README.md | 3 +++
1 file changed, 3 insertions(+)
create mode 100644 README.md
```