LABORATORIO 2 – GESTIÓN DE VERSIONES

Profesores:
Henry Umaña
Jairo Aponte
Liliana Olarte

Facultad de Ingeniería de Sistemas e Industrial
Ingeniería de Software I
2025-II
Bogotá, Colombia

INTRODUCCIÓN

En el mundo de la ingeniería de software, la necesidad de una gestión eficiente de las distintas versiones de un proyecto ha sido una constante. Esta necesidad ha sido satisfecha de manera sobresaliente por los gestores de versiones, herramientas que desempeñan un papel crucial en el control, seguimiento y colaboración en el desarrollo de software. Su importancia radica en su capacidad para facilitar la colaboración entre desarrolladores, permitir la reversión y recuperación de versiones anteriores en caso de problemas, rastrear cambios para mantener la responsabilidad y posibilitar la creación de ramas para la experimentación de nuevas características, factores que, en conjunto, contribuyen significativamente a la eficiencia y calidad en el ciclo de desarrollo de software.

GitHub es la elección predilecta para esta tarea debido a su sólido sistema de control de versiones, permitiendo un seguimiento preciso de modificaciones, lo que es esencial para mantener un historial completo y revertir a versiones anteriores en caso de problemas. Además, facilita la colaboración efectiva entre desarrolladores, lo que es esencial para el trabajo en equipo en el laboratorio. Por último, su integración continua y herramientas de gestión de proyectos hacen que sea una plataforma completa para abordar proyectos de desarrollo de software en un entorno de laboratorio.

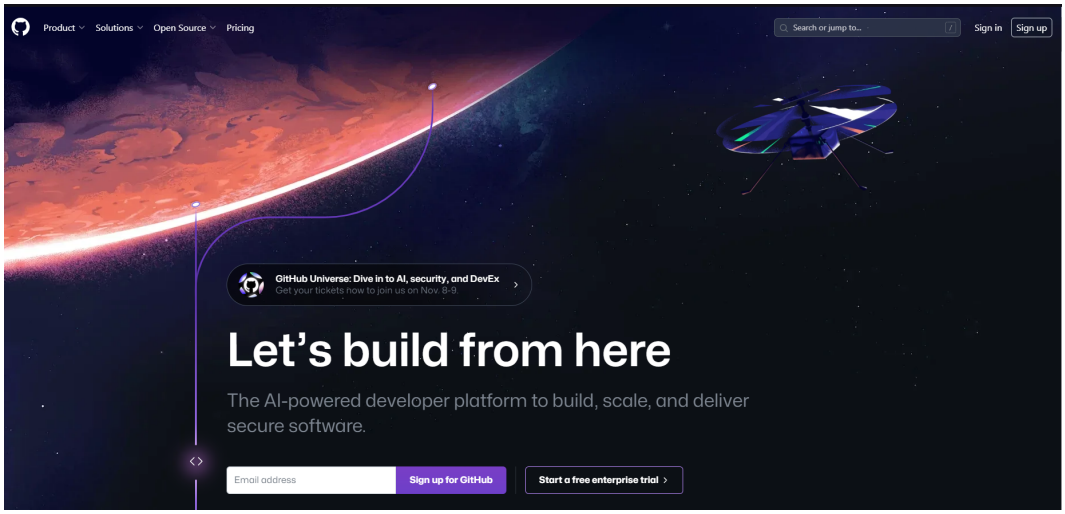
OBJETIVOS

1. Adquirir conocimientos sólidos sobre el control de versiones y colaboración efectiva utilizando GitHub.
2. Dominar la gestión de proyectos, incluyendo la organización de tareas y la configuración de flujos de trabajo eficientes en GitHub.
3. Desarrollar habilidades prácticas necesarias para el trabajo en desarrollo de software, incluyendo el uso de ramas, colaboración en equipos y la comprensión de conceptos clave de control de versiones.

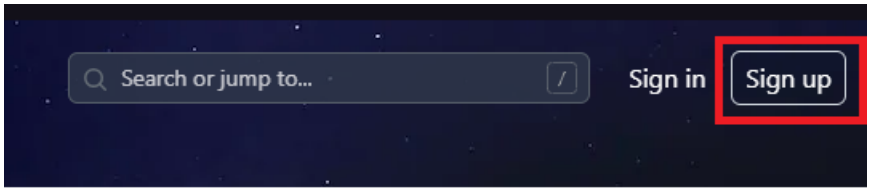
PREPARACIÓN DEL ENTORNO

Para poder realizar el laboratorio de manera satisfactoria, TODOS los integrantes del grupo deben contar con una cuenta activa de GitHub. Y la instalación de git en su ordenador. Para instalar el gestor realice el paso a paso del video adjunto. En el caso de no contar con una cuenta de Git Hub, siga los siguientes pasos:

Abre tu navegador web e ingresa a la página principal de GitHub.



En la página de inicio de GitHub, verás un botón verde llamado "Sign up" (Registrarse) en la esquina superior derecha. Haz clic en él.

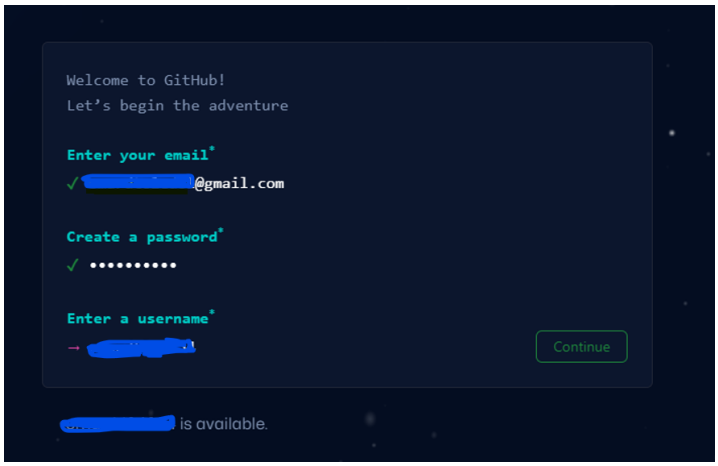


A continuación, se te redirigirá a la página de registro. Deberás proporcionar la siguiente información:

Username (Nombre de usuario): Elige un nombre de usuario único que se utilizará para identificarte en GitHub.

Email address (Dirección de correo electrónico): Proporciona una dirección de correo electrónico válida.

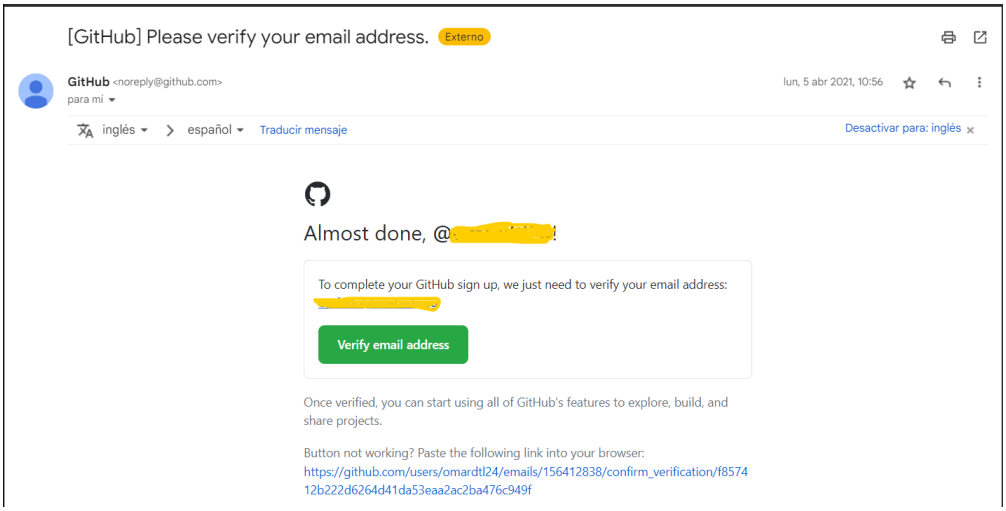
Password (Contraseña): Crea una contraseña segura.



También tienes la opción de registrarte usando tu cuenta de Google o tu cuenta de Microsoft si prefieres no crear un nombre de usuario y contraseña separados.

Después de completar los campos requeridos, haz clic en el botón verde "Sign up for GitHub" (Registrarse en GitHub).

GitHub te enviará un correo electrónico de verificación a la dirección de correo que proporcionaste. Abre tu correo electrónico y sigue las instrucciones para verificar tu dirección.



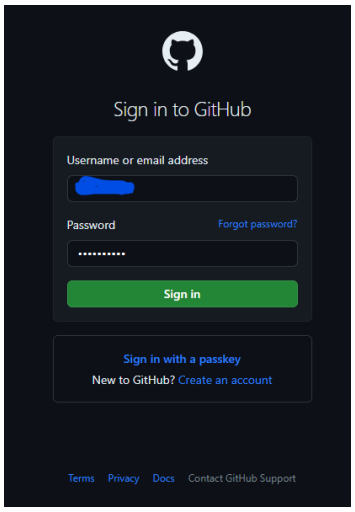
Una vez que hayas verificado tu dirección de correo electrónico, habrás creado tu cuenta en GitHub.

DESARROLLO

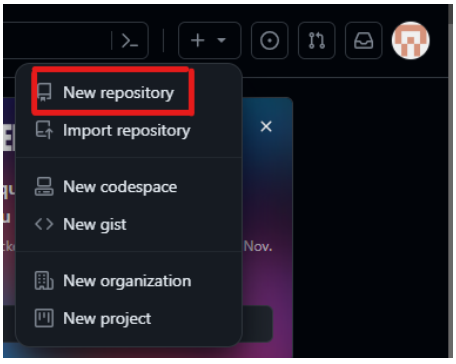
a. Creación de Repositorio Remoto

Para empezar a realizar el laboratorio, es necesario de disponer de un repositorio remoto. Solo UN integrante por grupo debe realizar el paso a paso que está asociado a esta parte del laboratorio (El resto del equipo simplemente será agregado al mismo para realizar el trabajo). Para crear un repositorio remoto, se debe hacer lo siguiente:

Inicia sesión en tu cuenta de GitHub.



Una vez que hayas iniciado sesión, estarás en tu página de inicio. En la parte superior derecha, encontrarás un ícono de signo más (+). Haz clic en él y selecciona "New repository" (Nuevo repositorio).



En la página "Create a new repository", deberás completar la información del repositorio:

Repository name (Nombre del repositorio): Ingresa un nombre único para tu repositorio.

Description (Descripción): Proporciona una breve descripción de tu proyecto.

Public or Private (Público o Privado): Selecciona si deseas que tu repositorio sea público (visible para todos) o privado (acceso restringido). (En este laboratorio se requiere que el repositorio permanezca público).

Initialize this repository with a README (Inicializar este repositorio con un README): Esta opción te permite crear un archivo README para la documentación del proyecto. (Para nuestro laboratorio, crearemos este archivo README y deberá contener el nombre del equipo y los nombres de los integrantes del mismo seguidos de sus username en GitHub).

Ejemplo del Contenido de README:

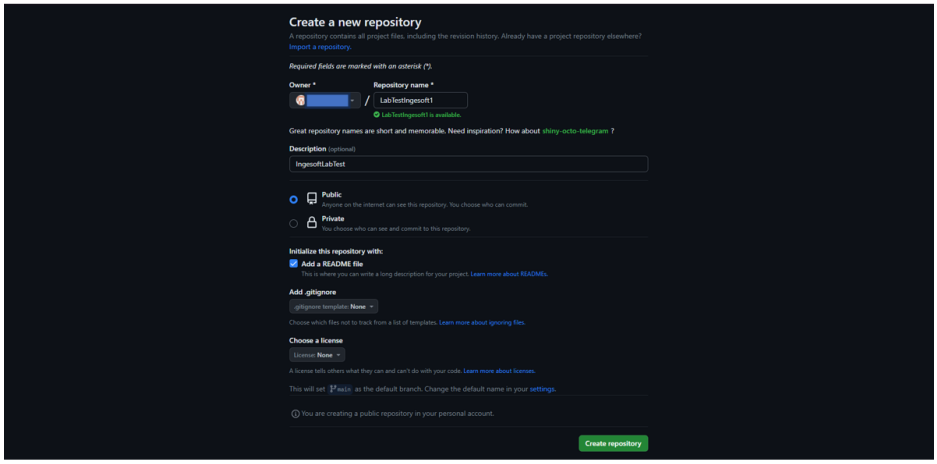
Team Name: El Clan del Bug

Team Members: Jaime Hernández – jher1234

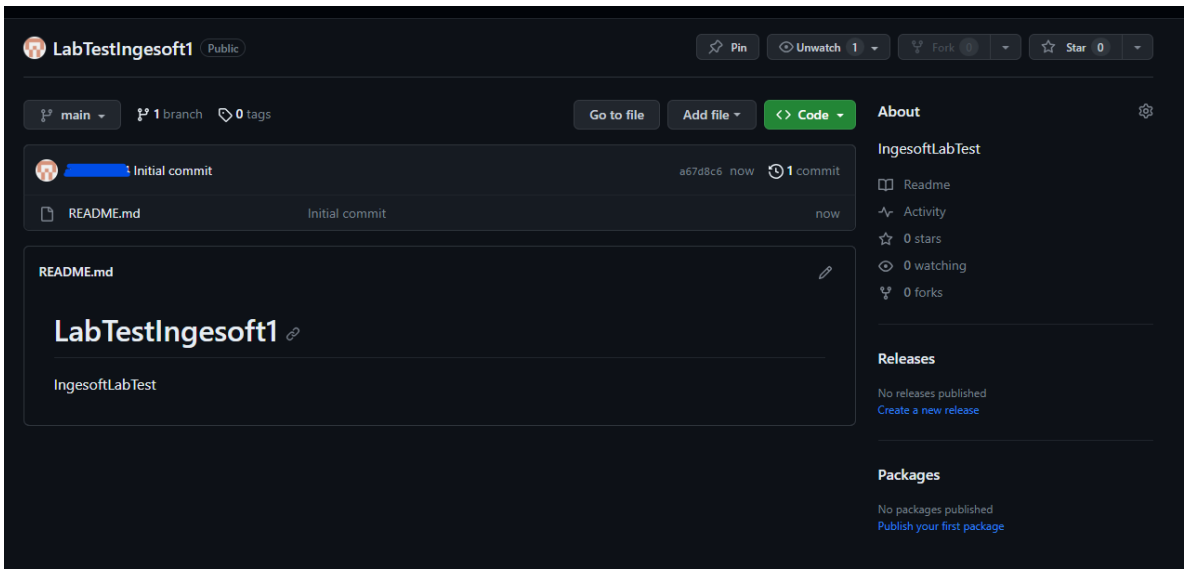
Lucía Vargas – luvar_dev

Add .gitignore: Puedes seleccionar un archivo .gitignore predeterminado según el lenguaje de programación que estés utilizando para evitar que se rastreen archivos innecesarios.

Add a license: Si deseas, puedes elegir una licencia para tu proyecto. (En este laboratorio se dejar este campo en None)

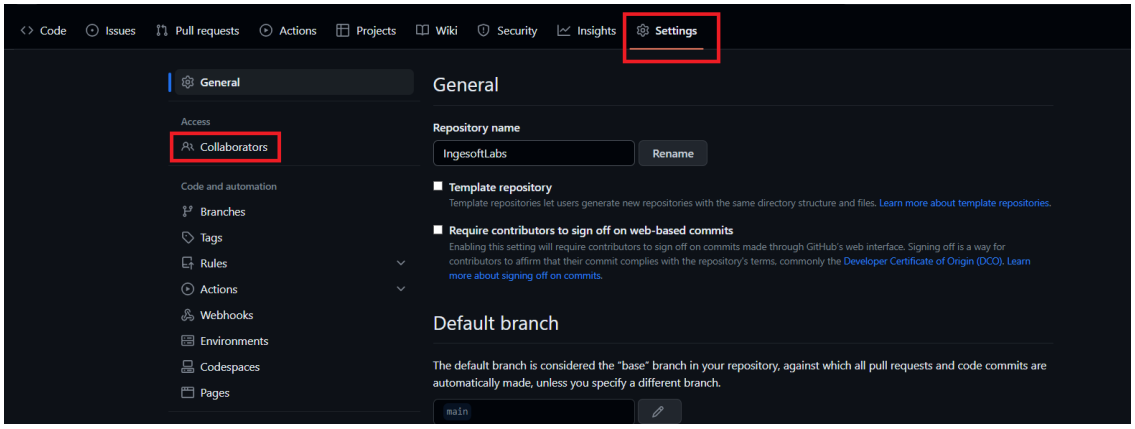


Una vez que hayas completado los detalles, haz clic en el botón verde "Create repository" (Crear repositorio).



Ya con el repositorio creado, es necesario incluir a todos los integrantes del equipo. Para ellos, es necesario seguir el siguiente paso a paso:

En la barra lateral izquierda de la página de configuración del repositorio, encontrarás la opción "Colaboradores" (Collaborators). Haz clic en ella.

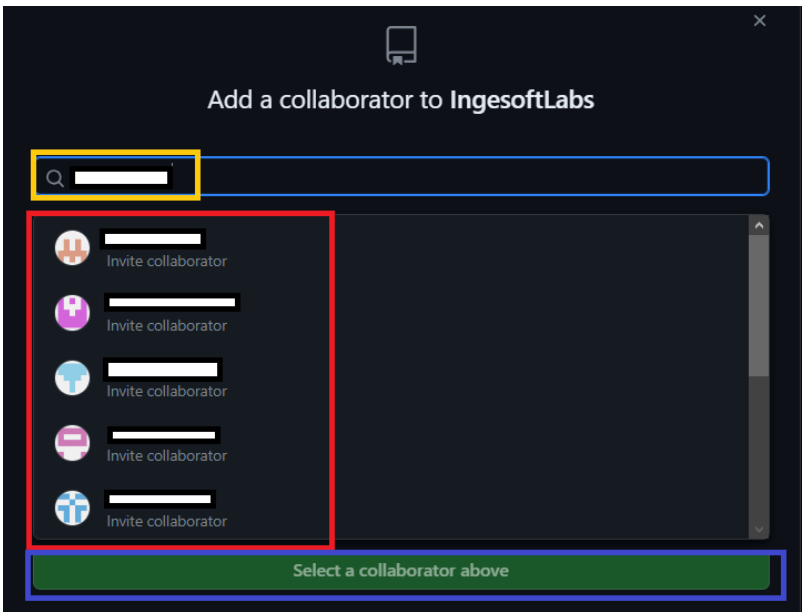


En la página "Colaboradores", verás un cuadro de búsqueda. Aquí puedes buscar a la persona que deseas agregar como colaborador.

Escribe el nombre de usuario de GitHub de la persona en el cuadro de búsqueda.

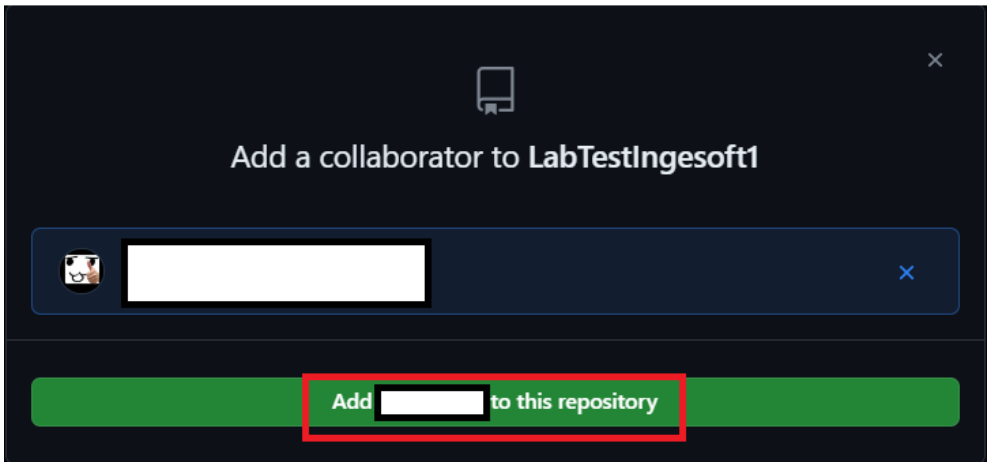
Haz clic en el nombre de usuario cuando aparezca en la lista de resultados.

Aparecerá una ventana emergente que te pedirá confirmar la colaboración de esta persona en el repositorio.



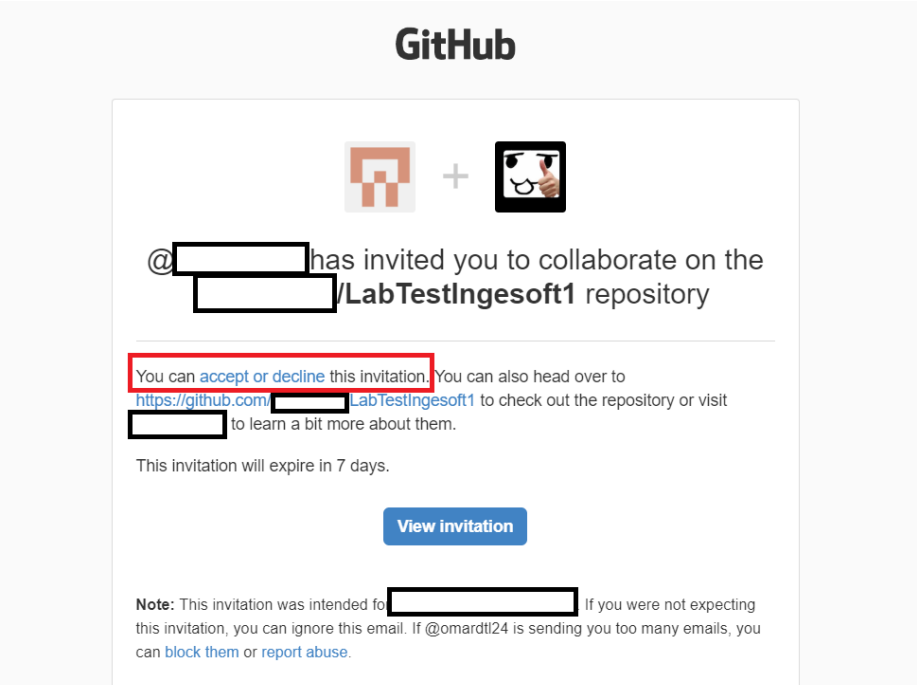
Puedes seleccionar los permisos que deseas otorgarles (lectura o escritura). Si solo deseas darles acceso de lectura, marca la casilla "Read" en lugar de "Write".

Haz clic en "Add [nombre de usuario] to [nombre del repositorio]".

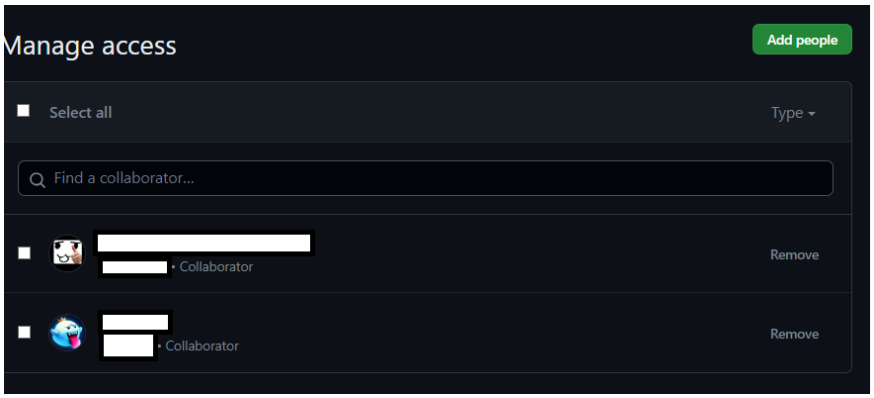


GitHub enviará una invitación a la persona para que colabore en el repositorio.

La persona invitada recibirá un correo electrónico notificándoles sobre la invitación.

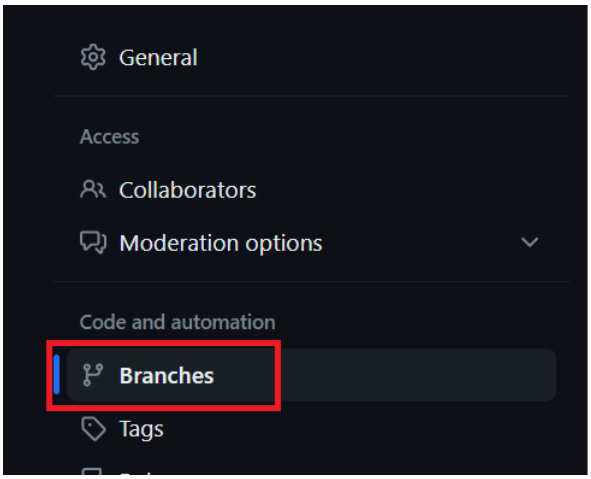


La persona invitada debe aceptar la invitación a través del enlace proporcionado en el correo electrónico o yendo a la pestaña "Invitaciones" (Invitations) en su página de inicio de GitHub.

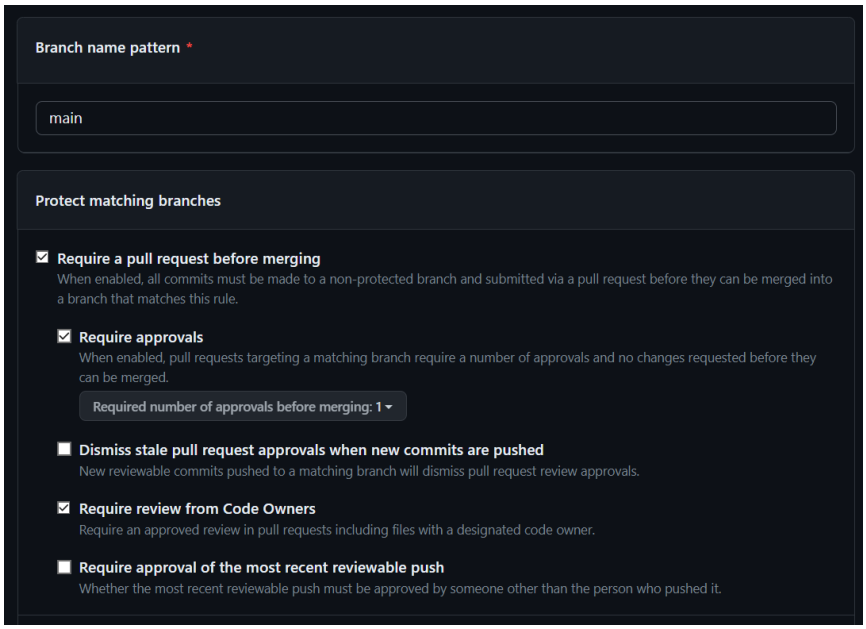


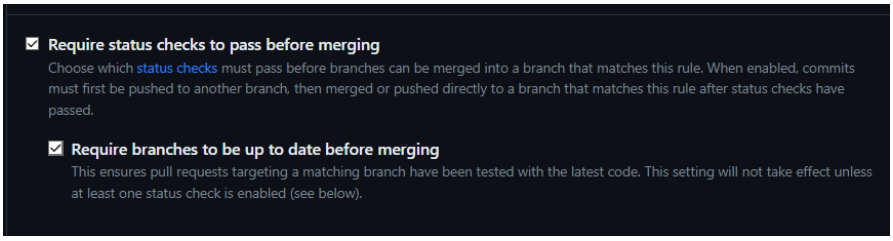
El siguiente paso es configurar la protección de la rama principal (main para nuestro caso). Esto se realiza con el propósito de permitir que solo se acepte el código de la más alta calidad y sea evaluado por los líderes de cada aspecto técnico del equipo. Para ello se seguirán los siguientes pasos:

El dueño del repositorio se dirige a la sección de configuración; y luego a la sección de ramas.



Posteriormente se configura en la rama principal los siguientes aspectos:





Finalmente, se le aplica *create* (En el caso de ya contar con la rama, esta no se duplicará)

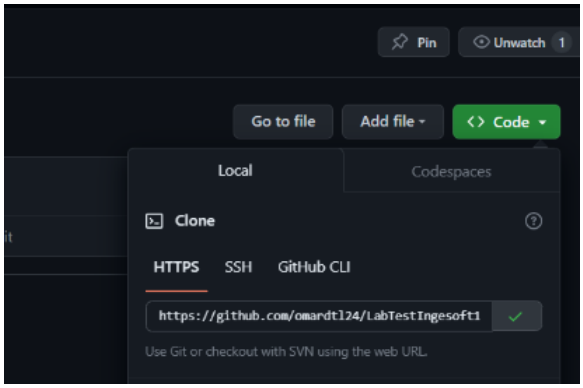
b. Conexión y Configuraciones Previas

El primer paso es que todos los integrantes del equipo conecten su entorno local de git al repositorio remoto. Para ello, se deben seguir los siguientes pasos.

Abre una terminal en tu computadora y ejecuta el siguiente comando, reemplazando <URL del repositorio> con la URL del repositorio remoto en GitHub:

git clone <URL del repositorio>

Esto descargará una copia del repositorio remoto en tu máquina local.




```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab
$ git clone https://github.com/omardtl24/LabTestIngesoft1.git
Cloning into 'LabTestIngesoft1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Conecta tu repositorio local con el remoto:

Entra al directorio de tu repositorio local con el comando cd turepositorio (reemplaza "turepositorio" con el nombre de tu repositorio).

Luego, configura Git para conectarse con tu repositorio remoto de GitHub utilizando el siguiente comando, donde <nombre_remoto> es un nombre que le asignas a tu repositorio remoto (por ejemplo, "origin") y <URL_del_repositorio> es la URL de tu repositorio de GitHub:

git remote add <nombre_remoto> <URL_del_repositorio>

Verifica que la conexión se haya establecido correctamente con el comando:

git remote -v

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab
$ cd LabTestIngesoft1

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestI
ngesoft1 (main)
$ git remote add GitLab https://github.com/omardtl24/LabTestIngesoft1.git

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git remote -v
GitLab  https://github.com/omardtl24/LabTestIngesoft1.git (fetch)
GitLab  https://github.com/omardtl24/LabTestIngesoft1.git (push)
origin  https://github.com/omardtl24/LabTestIngesoft1.git (fetch)
origin  https://github.com/omardtl24/LabTestIngesoft1.git (push)
```

En Git, tu nombre de usuario y dirección de correo electrónico se utilizan para registrar la autoría de tus confirmaciones (commits). Para configurarlos, utiliza los siguientes comandos:

git config --global user.name "Tu Nombre"

git config --global user.email "tu@email.com"

Para visualizar que hayan quedado bien registrados, utiliza los siguientes comandos:

git config --global user.name

git config --global user.email

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git config --global user.name omardtl24

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git config --global user.email otoledo@unal.edu.co

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git config --global user.name
omardtl24

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git config --global user.email
otoledo@unal.edu.co
```

c. Comandos y acciones

El primer comando a analizar es aquel que permite contrastar las diferencias entre las diferentes versiones que se van definiendo y confirmando en el repositorio local. Este es:

git status

En el caso de no existir diferencias entre las versiones confirmadas, el mensaje marcado es que el árbol de trabajo está limpio. En caso contrario, se indicará que archivos se han agregado, modificado o eliminado en contraste al último código versionado.

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   README.md

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

El segundo comando a revisar es aquel que permite agregar archivos a la zona de preparación para ser correctamente versionados. No genera ningún tipo de mensaje, para verificar que los cambios se agregaron se debe aplicar el comando git status. Este comando tiene diferentes versiones de acuerdo a lo que se necesite agregar:

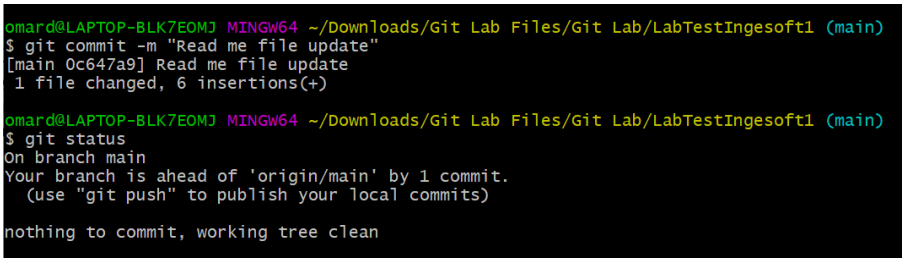
git add [filename] (Agrega el archivo puntual que se referencia)

git add . (Agrega todos los archivos que hayan sido identificados)

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git add README.md

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```

El tercer comando a analizar es aquel que nos permite versionar los cambios ya agregados. Este tiene diversas variaciones pero nos centraremos únicamente en una. Para verificar que los cambios se versionaron se debe aplicar el comando git status (Los cambios ya no deben ser visibles).

git commit -m "[Commit Message]"



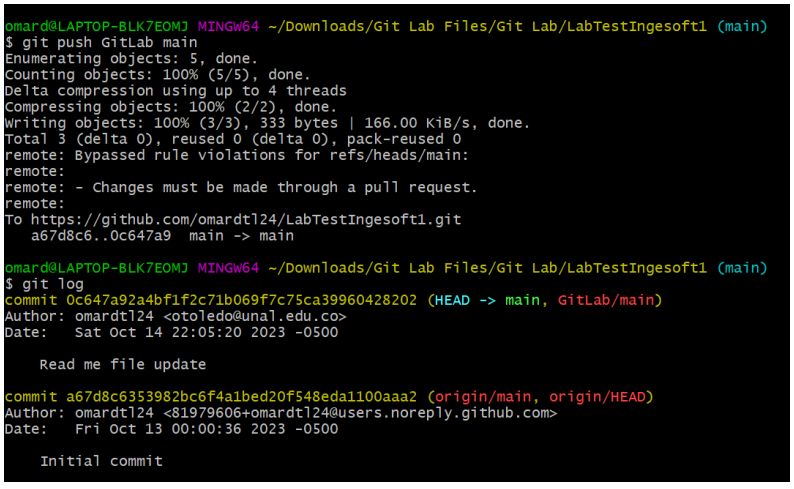
El siguiente paso es ser capaz de ver el historial de cambios realizados por el usuario. El comando a presentar se encarga de generar el historial de diferentes versiones con su respectivo Hash de identificación.

git log

Ya con los diferentes cambios manejados en el entorno local, es necesario enviarlos al repositorio remoto. Para ello utilizaremos el siguiente comando

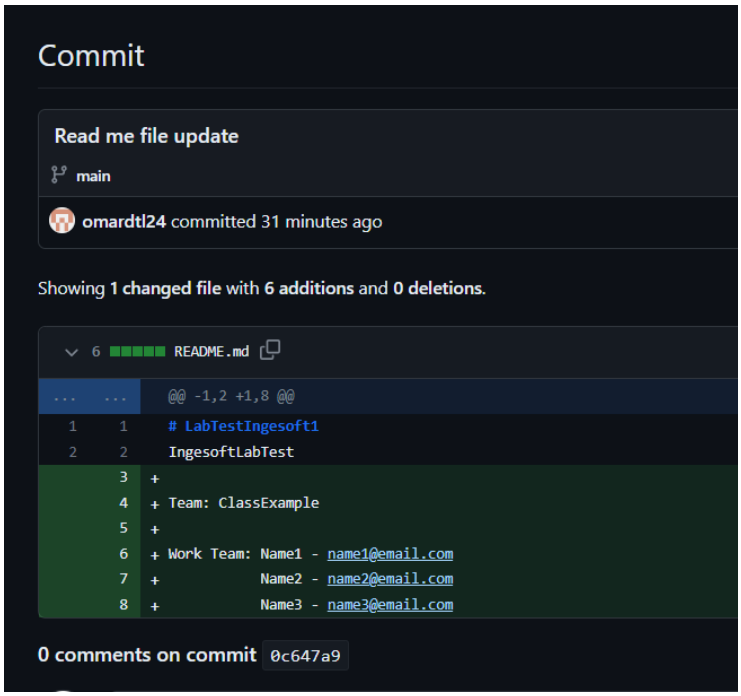
git push <repositorio-remoto> <rama-local>:<rama-remota>

git push <repositorio-remoto> <rama-remota>

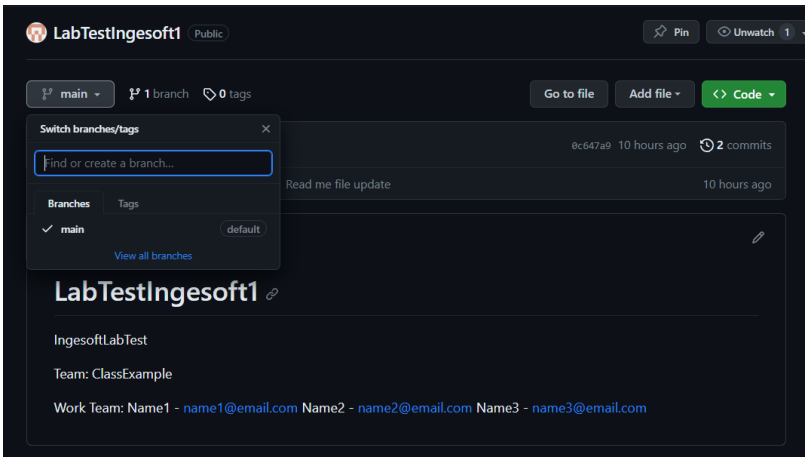


En este ejemplo, se realiza una advertencia de que los cambios a la rama main deben realizarse a través de un Pull Request (El cual se nombrará más adelante), aún así el cambio fue aceptado dado que lo realizó la cuenta asociada como creadora del repositorio.

En el entorno de GitHub se pueden ver los cambios realizados y la evolución de los diferentes archivos:



El siguiente aspecto a trabajar son las ramas, una rama es una línea de desarrollo separada que permite trabajar en características o modificaciones de un proyecto de software sin afectar la rama principal. Para ser capaz de visualizar las ramas asociadas a un repositorio remoto, podemos hacerlo a través de la interfaz de GitHub.



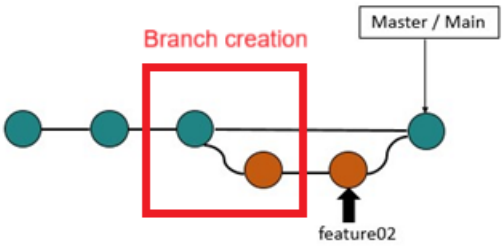
En el entorno de github local, también es viable realizar un manejo de ramas bastante eficiente; para ello, contamos con una serie de comandos que nos permite visualizar, crear y desplazarme entre las diferentes ramas.

El primer aspecto para analizar es la capacidad de visualizar las diferentes ramas del proyecto. Esto se logra con el siguiente comando que lista las diferentes ramas existentes y marca en verde aquella en la que se esté trabajando.

git branch

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git branch
* main
```

El segundo aspecto para analizar es la capacidad de crear una rama en un repositorio local. Para ello, debemos tener claro que, al crear una rama, estamos partiendo de una versión o momento específico de una rama previa (Que en muchos casos es la rama principal). Eso puede ser evidenciado de manera más clara en la siguiente imagen:



El comando que me permite esto es una variación de aquel que me permite visualizar las ramas de un repositorio local. Es importante tener en cuenta que al momento de aplicar el próximo comando a mostrar, se creará una rama a partir de la última en la que se esté ubicado:

git branch <nombre de nueva rama>

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git branch
* main

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git branch testbranch

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git branch
* main
  testbranch
```

En este ejemplo, podemos evidenciar que se creó una rama denominada “testbranch” a partir de la rama principal main.

El tercer aspecto para analizar es la capacidad de desplazarse entre las ramas de un repositorio local. Es una tarea muy sencilla y se aplica con el siguiente comando:

git checkout <nombre de rama a desplazarse>

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git branch
* main
  testbranch

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git checkout testbranch
Switched to branch 'testbranch'

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (testbranch)
$ git branch
  main
* testbranch
```

Todos los cambios realizados a partir de ese punto se reflejarán sobre la rama “testbranch”; por lo que todo el proceso de agregar y versionar archivos se realizarán sobre la misma. A continuación, se muestra un flujo corto de trabajo que se realizó sobre la rama “testbranch”.

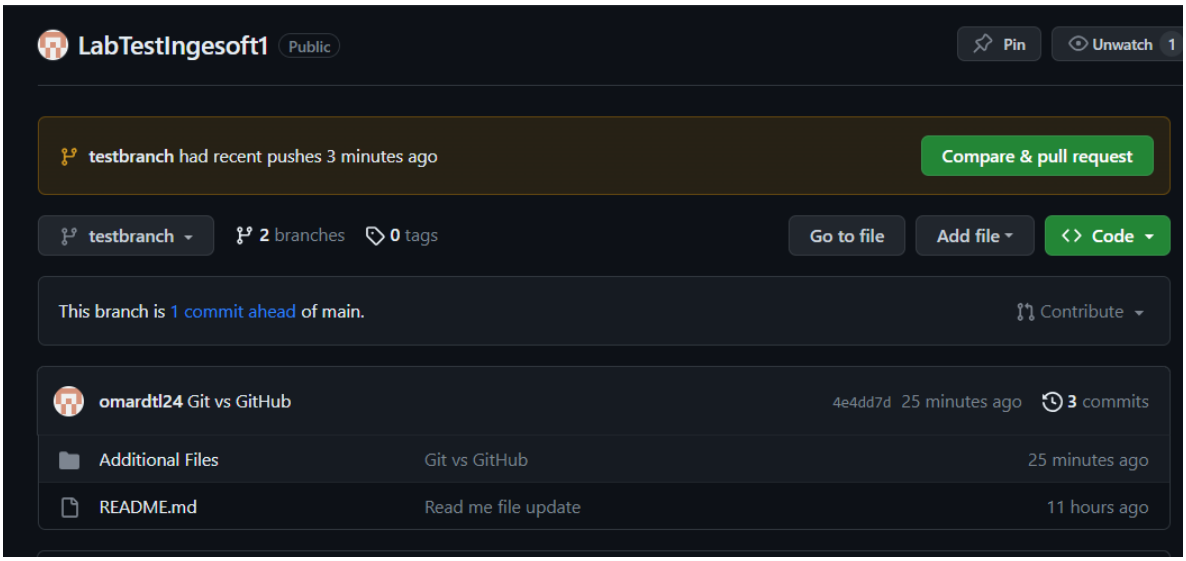
```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (testbranch)
$ git status
On branch testbranch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Additional Files/

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

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (testbranch)
$ git add .

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (testbranch)
$ git commit -m "Git vs GitHub"
[testbranch 4e4dd7d] Git vs GitHub
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Additional Files/Git_GitHub_comparison.png

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (testbranch)
$ git log
commit 4e4dd7d86d0c15ce468cd75ddc05afc2d78744c5 (HEAD -> testbranch)
Author: omardtl24 <otoledo@unal.edu.co>
Date:   Sun Oct 15 09:23:54 2023 -0500

    Git vs GitHub

commit 0c647a92a4bf1f2c71b069f7c75ca39960428202 (GitLab/main, main)
Author: omardtl24 <otoledo@unal.edu.co>
Date:   Sat Oct 14 22:05:20 2023 -0500

    Read me file update

commit a67d8c6353982bc6f4a1bed20f548eda1100aaa2 (origin/main, origin/HEAD)
Author: omardtl24 <81979606+omardtl24@users.noreply.github.com>
Date:   Fri Oct 13 00:00:36 2023 -0500

    Initial commit
```

Por último, buscamos subir el código realizado en esta rama, a una rama del mismo nombre en el repositorio remoto en GitHub. Para ello, utilizaremos el comando push,

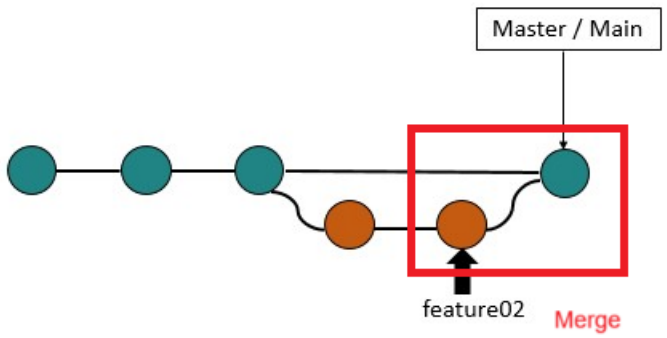
teniendo en cuenta que el nombre de la rama será aquella que será llevada al repositorio remoto.

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (testbranch)
$ git push GitLab testbranch
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 465.09 KiB | 23.25 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'testbranch' on GitHub by visiting:
remote:      https://github.com/omardtl24/LabTestIngesoft1/pull/new/testbranch
remote:
To https://github.com/omardtl24/LabTestIngesoft1.git
 * [new branch]      testbranch -> testbranch
```

En GitHub quedará reflejada la nueva rama con los cambios asociados a la misma.



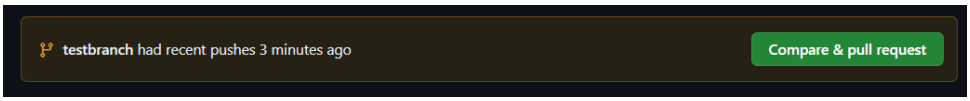
El último aspecto importante a tener en cuenta en el manejo de ramas es la capacidad de regresar al flujo principal de trabajo. Este proceso es conocido como merge.

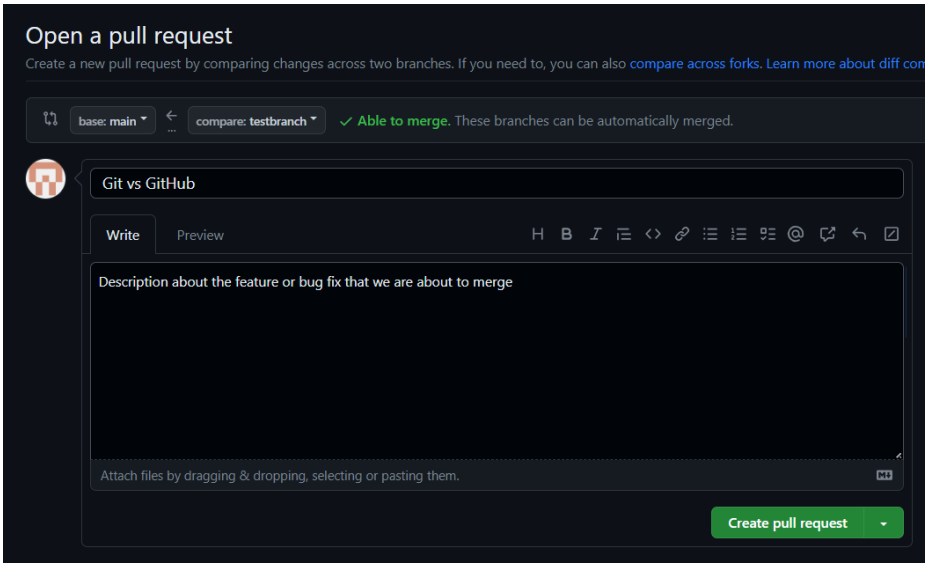


Realizar un proceso de merge es aquel que puede traer más problemas a la hora de manejar un gestor de versiones. Esto se debe a que pueden surgir conflictos

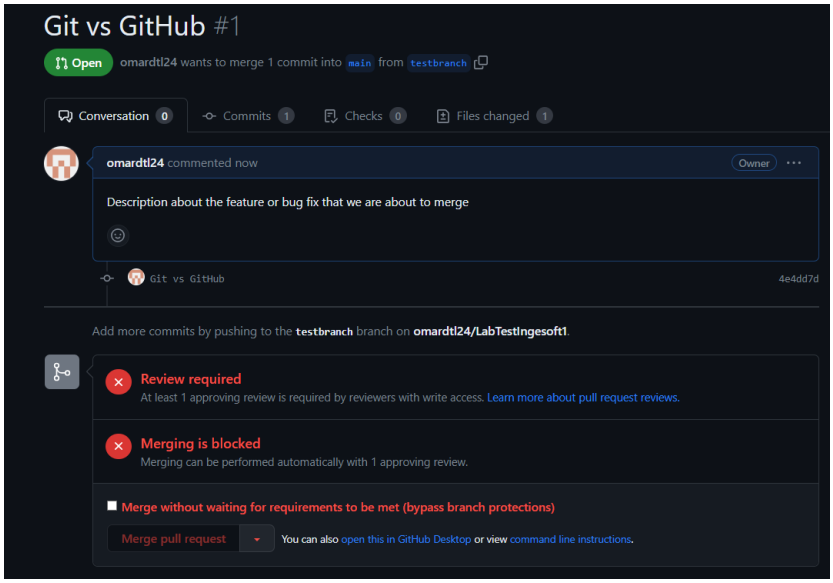
dentro del desarrollo del código. En este paso a paso, vamos a suponer que los conflictos ya fueron resueltos para que no sea un proceso demasiado extenso.

Para realizar el merge en un repositorio remoto, debemos realizar lo que se denomina un "pull request". Es decir, una petición para poder llevar los cambios de una rama a otra. Para ello, simplemente indicamos en el botón Compare & pull request y llenamos la información asociada al cambio de deseamos unir a la rama. (Nótese que la rama a la que se unirán los cambios se denominará *base*, mientras que la rama de donde se traerán los cambios se denomina *compare*).

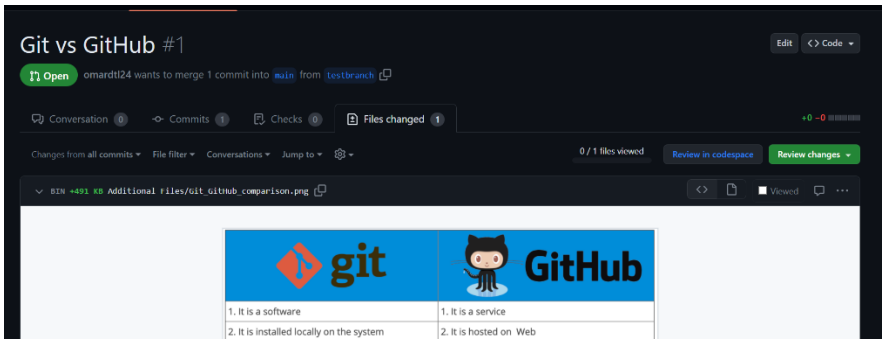


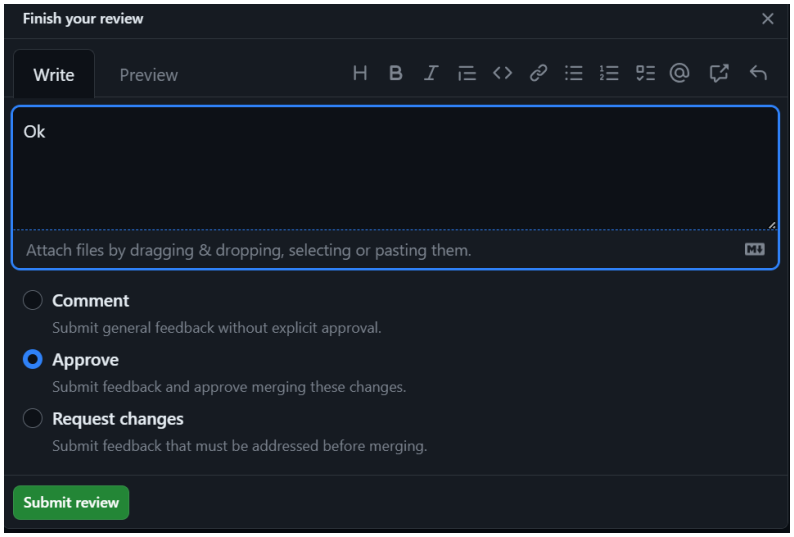


Después de realizar el pull request, se recibirá el siguiente mensaje:

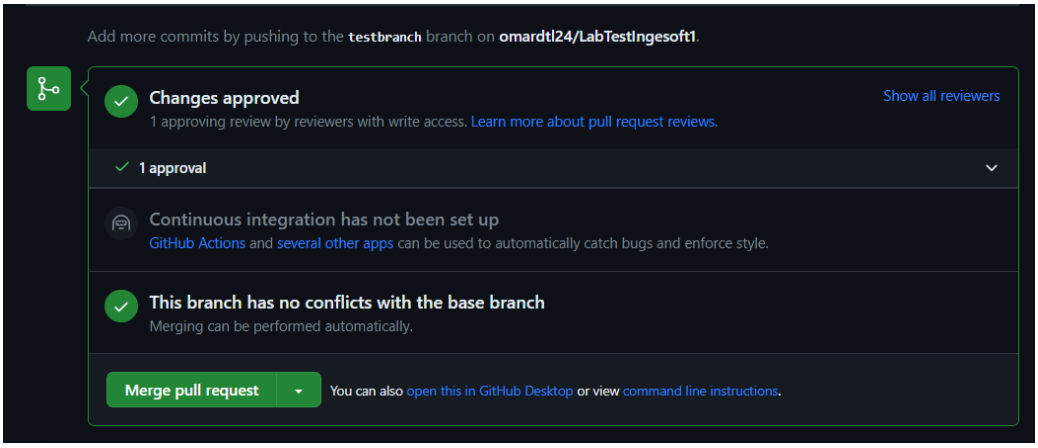


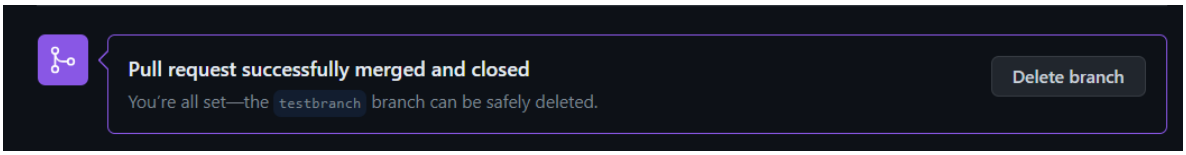
Este no implica que haya sucedido algún error, este simplemente requiere que otro integrante del equipo (Usualmente el líder técnico) evalúe los cambios antes de ser llevados a la rama main (Se definió este mecanismo de seguridad en la configuración del repositorio para garantizar que se mantenga en la rama principal el código de más alta calidad).





Después de finalizado el proceso, el merge a la rama principal es autorizado y la rama auxiliar ya puede ser eliminada (Por buena práctica, es preferible eliminar ramas a las que ya se les haya hecho merge).





Para realizar el merge en un repositorio local, se deben usar los comandos git merge y git checkout de la manera adecuada (Tenga en cuenta que hacer este proceso de forma incorrecta puede afectar el origen y destino de los cambios entre las ramas). El uso correcto de los comandos es el siguiente:

git checkout <Rama destino de cambios (base)>

git merge <Rama origen de cambios (compare)>

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (testbranch)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git merge testbranch
Updating 0c647a9..4e4dd7d
Fast-forward
 Additional Files/Git_GitHub_comparison.png | Bin 0 -> 502420 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Additional Files/Git_GitHub_comparison.png
```

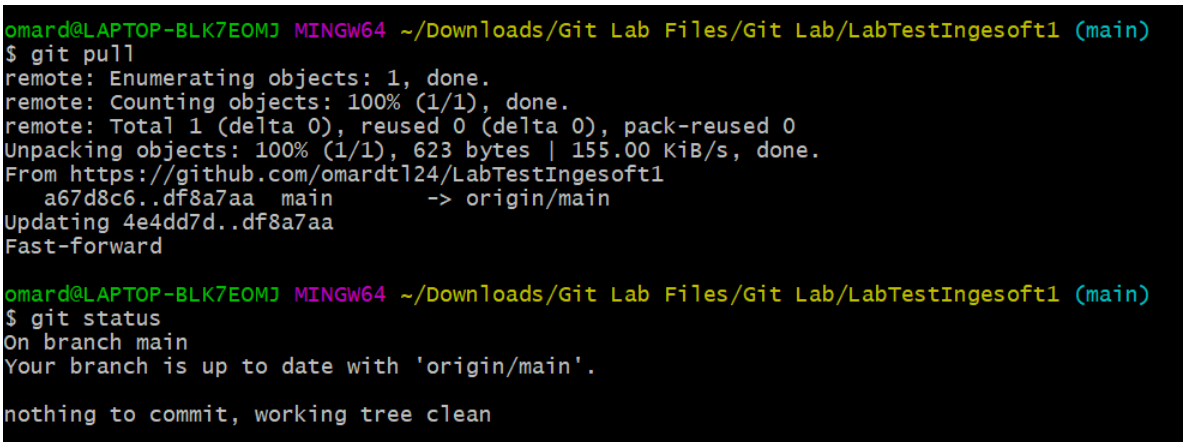
Después de realizado el proceso de merge, podemos eliminar la rama auxiliar con el siguiente comando (En el caso de no haber realizado adecuadamente los merge, este generará una advertencia):

git branch -d <Nombre de Rama a eliminar>

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git branch
* main
  testbranch

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git branch -d testbranch
Deleted branch testbranch (was 4e4dd7d).

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git branch
* main
```

Por último, para traer los cambios realizados al repositorio local desde el repositorio remoto, se debe usar el siguiente comando:

git pull <Repositorio Remoto>



Este puede generar conflictos si no se han sincronizado correctamente los repositorios de manera adecuada. Por consiguiente, es recomendable revisar ese aspecto antes de intentar traer cambios al repositorio local.

d. Flujo de Trabajo

Para el entregable del laboratorio, se tendrán en cuenta dos partes:

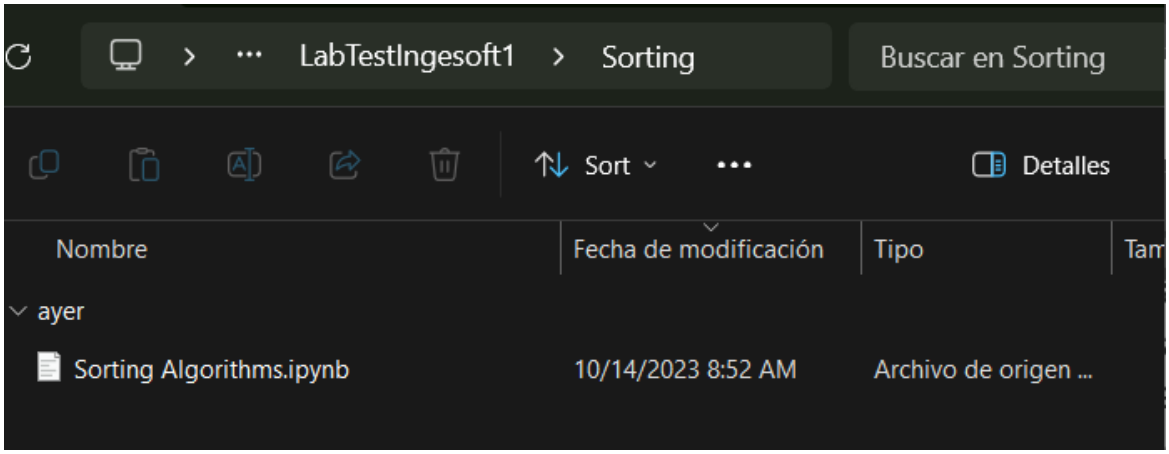
1. **Trabajo en Conjunto sobre el mismo archivo**
   Para esta primera parte, el equipo de trabajo participará en la construcción del Notebook de Python denominado "*Sorting Algorithms.ipynb*", en el adjunto a este documento, podrán encontrar las 7 diferentes versiones del archivo (Cada una siendo la continuación de la inmediatamente anterior).

   El paso a paso a seguir es el siguiente:

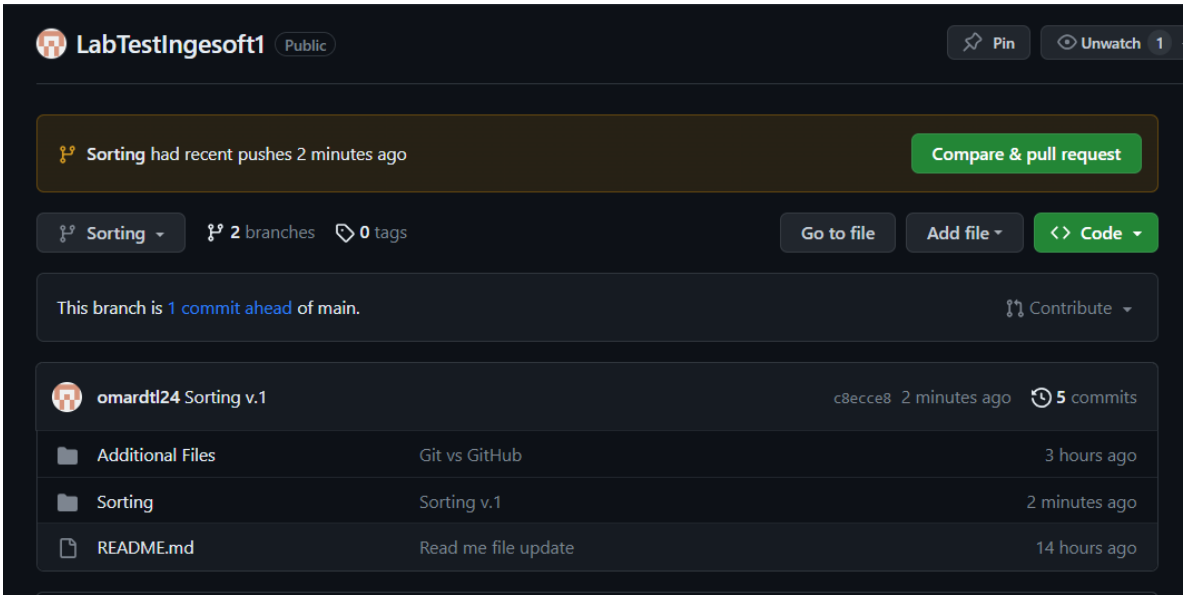
   I. Un integrante del grupo crea la rama Sorting a partir de la rama principal en su repositorio local, crea la carpeta Sort y dentro de esta deja la primera versión del archivo. Posteriormente versiona estos cambios y los envía al repositorio remoto.

Versión 1

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git branch Sorting

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git checkout Sorting
Switched to branch 'Sorting'

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git status
On branch Sorting
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Sorting/

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



```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git add .
warning: LF will be replaced by CRLF in Sorting/Sorting Algorithms.ipynb.
The file will have its original line endings in your working directory

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git commit -m "Sorting v.1"
[Sorting c8ecce8] Sorting v.1
 1 file changed, 69 insertions(+)
 create mode 100644 Sorting/Sorting Algorithms.ipynb

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git push GitLab Sorting
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 954 bytes | 954.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'Sorting' on GitHub by visiting:
remote:      https://github.com/omardtl24/LabTestIngesoft1/pull/new/Sorting
remote:
To https://github.com/omardtl24/LabTestIngesoft1.git
 * [new branch]      Sorting -> Sorting
```



II. De manera secuencial, los integrantes se ubican en la rama Sorting de su repositorio local (La crean si no cuentan con ella), realizan pull para traer la versión más actualizada del código y las ramas, actualizan el archivo de acuerdo con la versión que van generando (A manera de sugerencia, a la hora de realizar los diferentes commits, es preferible nombrarlos de acuerdo a la versión que se va trabajando); adicionalmente, es importante respetar la secuencia descrita para evitar conflictos. En el caso de que ya todos los integrantes del equipo hayan realizado su versión y no hayan alcanzado las 7 versiones del archivo, deben iniciar el ciclo nuevamente.

Versión 2

```
Reinaldo Toledo@LAPTOP-ROEKER19 MINGW64 ~/Desktop/Gitlabtest/LabTestIngesoft1 (main)
$ git pull origin Sorting
From https://github.com/omardtl24/LabTestIngesoft1
 * branch            Sorting    -> FETCH_HEAD
Updating df8a7aa..c8ecce8
Fast-forward
 Sorting/Sorting Algorithms.ipynb | 69 +++++++++++++++++++++++++++++++++++++++++
 1 file changed, 69 insertions(+)
 create mode 100644 Sorting/Sorting Algorithms.ipynb

Reinaldo Toledo@LAPTOP-ROEKER19 MINGW64 ~/Desktop/Gitlabtest/LabTestIngesoft1 (main)
$ git checkout Sorting
Switched to branch 'Sorting'
Your branch is up to date with 'origin/Sorting'.

Reinaldo Toledo@LAPTOP-ROEKER19 MINGW64 ~/Desktop/Gitlabtest/LabTestIngesoft1 (Sorting)
$ git status
On branch Sorting
Your branch is up to date with 'origin/Sorting'.

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:   Sorting/Sorting Algorithms.ipynb

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

Reinaldo Toledo@LAPTOP-ROEKER19 MINGW64 ~/Desktop/Gitlabtest/LabTestIngesoft1 (Sorting)
$ git add .
warning: in the working copy of 'Sorting/Sorting Algorithms.ipynb', LF will be replaced by CRLF the next
 time Git touches it

Reinaldo Toledo@LAPTOP-ROEKER19 MINGW64 ~/Desktop/Gitlabtest/LabTestIngesoft1 (Sorting)
$ git commit -m "Sorting v.2"
[Sorting 8628e09] Sorting v.2
 1 file changed, 56 insertions(+), 2 deletions(-)

Reinaldo Toledo@LAPTOP-ROEKER19 MINGW64 ~/Desktop/Gitlabtest/LabTestIngesoft1 (Sorting)
$ git push origin Sorting
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 627 bytes | 627.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/omardtl24/LabTestIngesoft1.git
   c8ecce8..8628e09  Sorting -> Sorting
```

Versión 3

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git pull origin main
From https://github.com/omardtl24/LabTestIngesoft1
 * branch            main       -> FETCH_HEAD
Already up to date.

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git pull origin Sorting
From https://github.com/omardtl24/LabTestIngesoft1
 * branch            Sorting    -> FETCH_HEAD
Updating df8a7aa..8628e09
Fast-forward
 Sorting/Sorting Algorithms.ipynb | 123 +++++++++++++++++++++++++++++++++++++++++
 1 file changed, 123 insertions(+)
 create mode 100644 Sorting/Sorting Algorithms.ipynb

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git checkout -b Sorting
Switched to a new branch 'Sorting'

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git status
On branch Sorting
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:   Sorting/Sorting Algorithms.ipynb

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

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git add .
warning: LF will be replaced by CRLF in Sorting/Sorting Algorithms.ipynb.
The file will have its original line endings in your working directory

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git commit -m "Sorting v.3"
[Sorting 4a1959f] Sorting v.3
 1 file changed, 45 insertions(+), 1 deletion(-)

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git push origin Sorting
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 619 bytes | 619.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/omardtl24/LabTestIngesoft1
   8628e09..4a1959f  Sorting -> Sorting
```

Versión 4

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git pull origin main
From https://github.com/omardtl24/LabTestIngesoft1
 * branch            main       -> FETCH_HEAD
Already up to date.

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git pull origin Sorting
From https://github.com/omardtl24/LabTestIngesoft1
 * branch            Sorting    -> FETCH_HEAD
Already up to date.

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git checkout Sorting
Already on 'Sorting'
M       Sorting/Sorting Algorithms.ipynb

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git status
On branch Sorting
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:   Sorting/Sorting Algorithms.ipynb

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

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git add .
warning: LF will be replaced by CRLF in Sorting/Sorting Algorithms.ipynb.
The file will have its original line endings in your working directory

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git commit -m "Sorting v.4"
[Sorting a7ec91f] Sorting v.4
 1 file changed, 37 insertions(+)

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git push origin Sorting
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 541 bytes | 541.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/omardtl24/LabTestIngesoft1
   4a1959f..a7ec91f  Sorting -> Sorting
```

Versión 5

```
Reinaldo Toledo@LAPTOP-R0EKER19 MINGW64 ~/Desktop/Gitlabtest/LabTestIngesoft1 (m
ain)
$ git checkout -b Sorting
Switched to a new branch 'Sorting'

Reinaldo Toledo@LAPTOP-R0EKER19 MINGW64 ~/Desktop/Gitlabtest/LabTestIngesoft1 (S
orting)
$ git pull origin Sorting
From https://github.com/omardtl24/LabTestIngesoft1
 * branch            Sorting    -> FETCH_HEAD
Updating df8a7aa..a7ec91f
Fast-forward
 Sorting/Sorting Algorithms.ipynb | 204 ++++++++++++++++++++++++++++++++++++++++
 1 file changed, 204 insertions(+)
 create mode 100644 Sorting/Sorting Algorithms.ipynb

Reinaldo Toledo@LAPTOP-R0EKER19 MINGW64 ~/Desktop/Gitlabtest/LabTestIngesoft1 (S
orting)
$ git status
On branch Sorting
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:   Sorting/Sorting Algorithms.ipynb

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

Reinaldo Toledo@LAPTOP-R0EKER19 MINGW64 ~/Desktop/Gitlabtest/LabTestIngesoft1 (Sorting)
$ git add .
warning: in the working copy of 'Sorting/Sorting Algorithms.ipynb', LF will be replaced by CRLF the next time Git touches it

Reinaldo Toledo@LAPTOP-R0EKER19 MINGW64 ~/Desktop/Gitlabtest/LabTestIngesoft1 (Sorting)
$ git commit -m "Sorting v.5"
[Sorting 3d9c9b3] Sorting v.5
 1 file changed, 43 insertions(+), 2 deletions(-)

Reinaldo Toledo@LAPTOP-R0EKER19 MINGW64 ~/Desktop/Gitlabtest/LabTestIngesoft1 (Sorting)
$ git push origin Sorting
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 547 bytes | 547.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/omardtl24/LabTestIngesoft1.git
   a7ec91f..3d9c9b3  Sorting -> Sorting
```

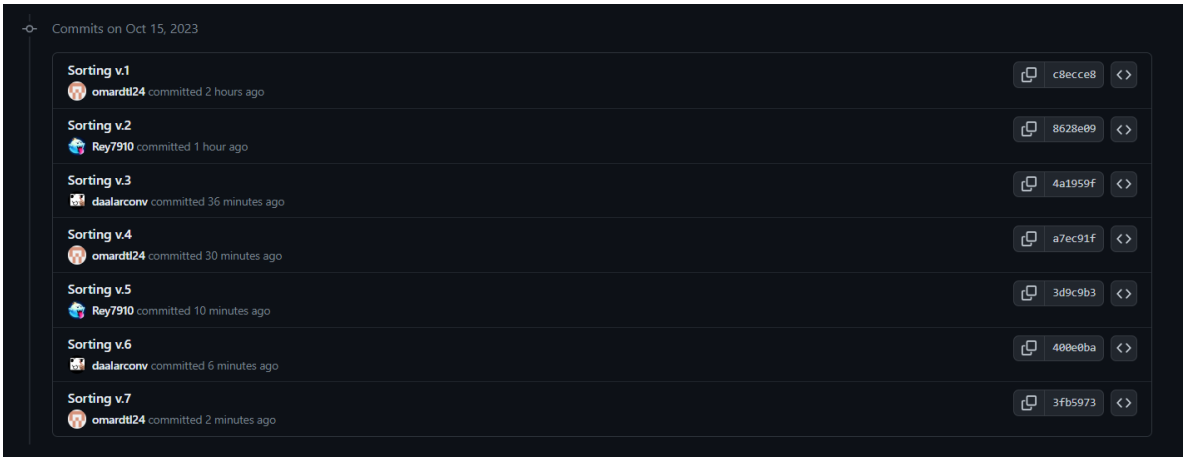
Versión 6

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git checkout Sorting
Switched to branch 'Sorting'

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git pull origin Sorting
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0
Unpacking objects: 100% (4/4), 527 bytes | 37.00 KiB/s, done.
From https://github.com/omardtl24/LabTestIngesoft1
 * branch            Sorting    -> FETCH_HEAD
   a7ec91f..3d9c9b3  Sorting    -> origin/Sorting
Updating a7ec91f..3d9c9b3
Fast-forward
 Sorting/Sorting Algorithms.ipynb | 45 ++++++++++++++++++++++++++++++++++++++++--
 1 file changed, 43 insertions(+), 2 deletions(-)

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git status
On branch Sorting
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:   Sorting/Sorting Algorithms.ipynb

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

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git add .
warning: LF will be replaced by CRLF in Sorting/Sorting Algorithms.ipynb.
The file will have its original line endings in your working directory

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git commit -m "Sorting v.6"
[Sorting 400e0ba] Sorting v.6
 1 file changed, 77 insertions(+), 10 deletions(-)

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git push origin Sorting
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.33 KiB | 1.33 MiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/omardtl24/LabTestIngesoft1
   3d9c9b3..400e0ba  Sorting -> Sorting
```

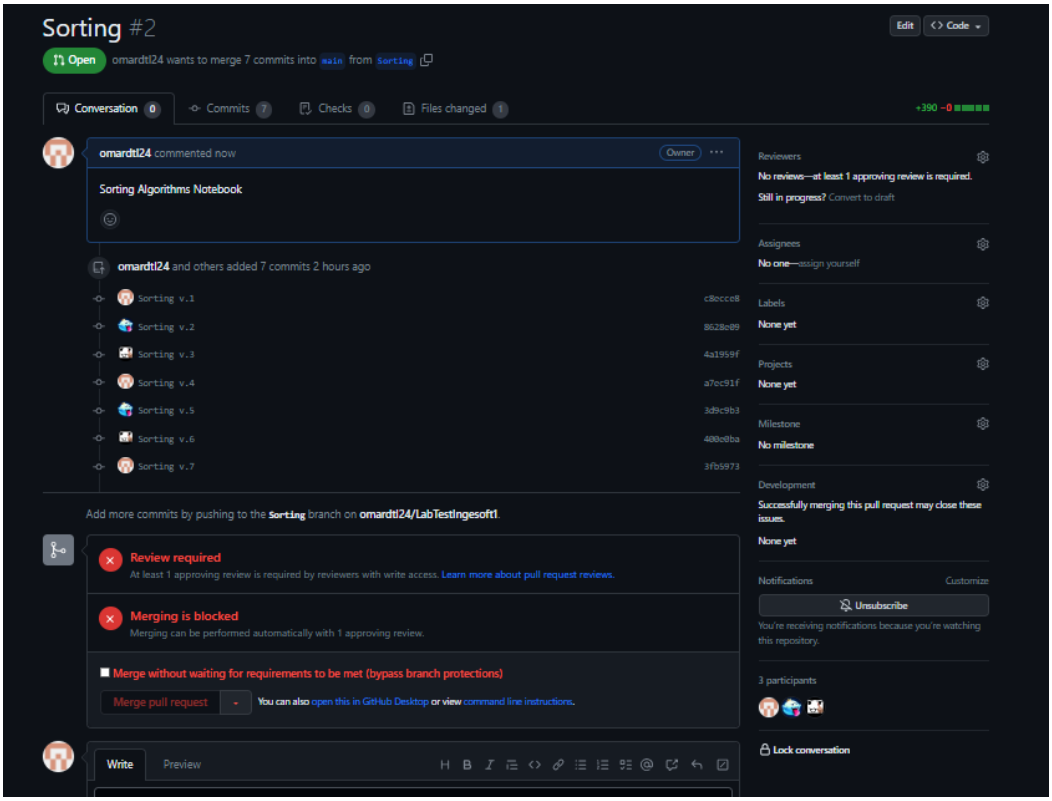
Versión 7

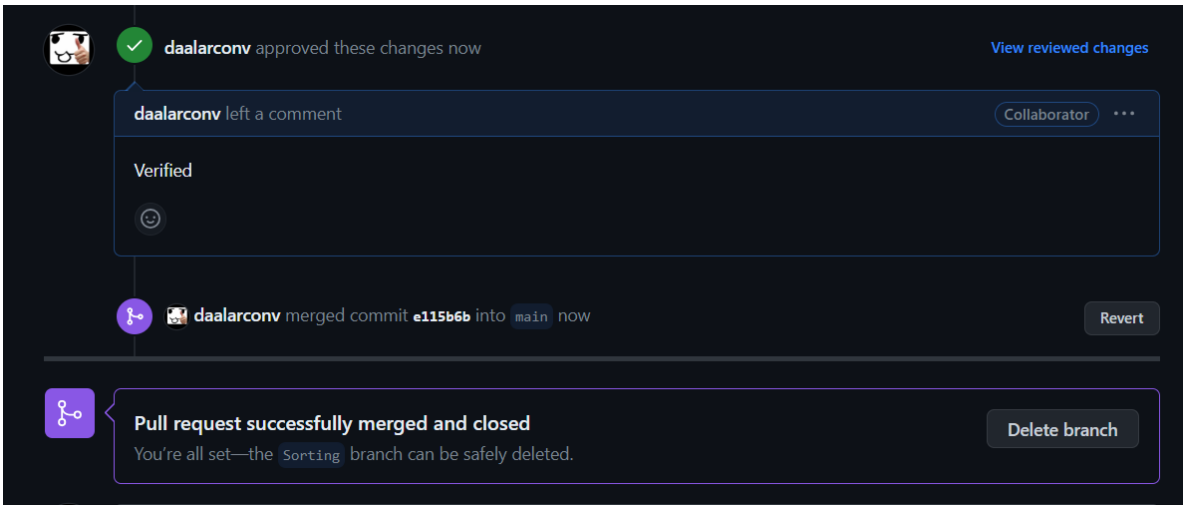
```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (main)
$ git checkout Sorting
Switched to branch 'Sorting'

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git pull origin Sorting
From https://github.com/omardtl24/LabTestIngesoft1
 * branch            Sorting    -> FETCH_HEAD
Already up to date.

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git status
On branch Sorting
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:   Sorting/Sorting Algorithms.ipynb

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

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git add .
warning: LF will be replaced by CRLF in Sorting/Sorting Algorithms.ipynb.
The file will have its original line endings in your working directory

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git commit -m "Sorting v.7"
[Sorting 3fb5973] Sorting v.7
 1 file changed, 95 insertions(+), 17 deletions(-)

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1 (Sorting)
$ git push origin Sorting
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 57.11 KiB | 9.52 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/omardtl24/LabTestIngesoft1
   400e0ba..3fb5973  Sorting -> Sorting
```

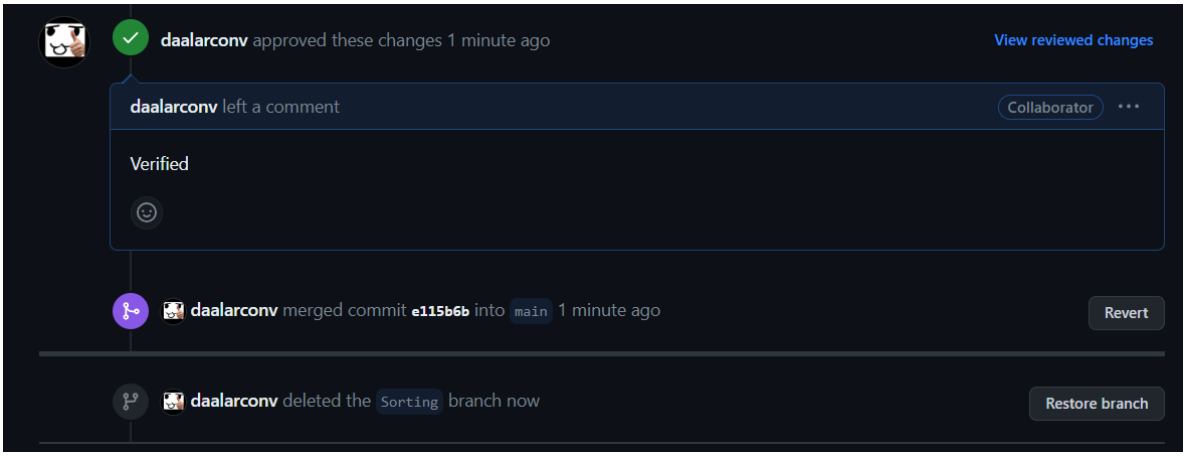
Si el proceso se hizo adecuadamente, el historial de versiones dentro de la rama Sorting del repositorio remoto se deberá ver algo así (Claramente con la participación de cada integrante del equipo):



III. Cuando ya se haya completado el versionamiento del archivo, se debe realizar el merge de la rama Sorting a la rama principal en el repositorio remoto. (Si el proceso se hizo adecuadamente, no existirán conflictos a la hora de hacer el merge). Al final deben eliminar la rama Sorting del repositorio remoto







2. **Trabajo individual sobre archivos independientes**
   El siguiente paso a paso, es individual para cada integrante del grupo, se basará en que cada uno creará una rama nueva en el repositorio remoto y hará un flujo de trabajo corto.

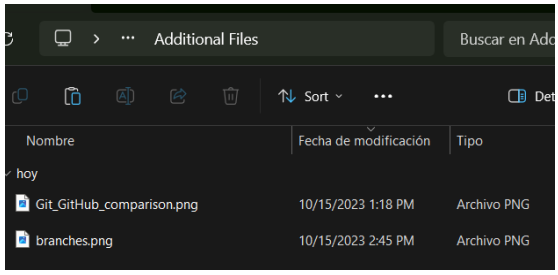
   1. Actualizar la rama principal del repositorio local con la información de la rama principal del repositorio remoto.

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1
 (main)
$ git pull origin main
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 636 bytes | 159.00 KiB/s, done.
From https://github.com/omardtl24/LabTestIngesoft1
 * branch            main       -> FETCH_HEAD
   df8a7aa..e115b6b  main       -> origin/main
Updating 3fb5973..e115b6b
Fast-forward

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1
 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

2. Crear una rama en el repositorio local llamada igual que el nombre de usuario del integrante en Github a partir de la rama principal.

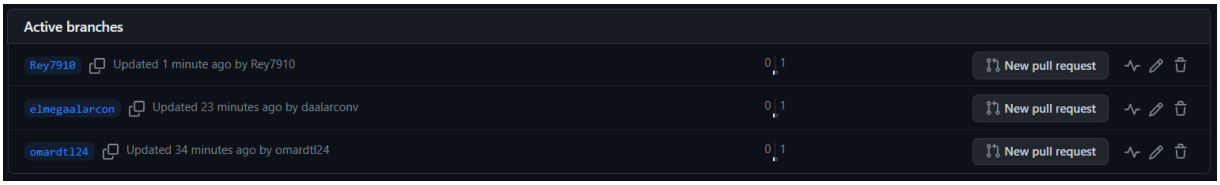
```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1
 (main)
$ git branch omardtl24

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1
 (main)
$ git checkout omardtl24
Switched to branch 'omardtl24'

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1
 (omardtl24)
$
```

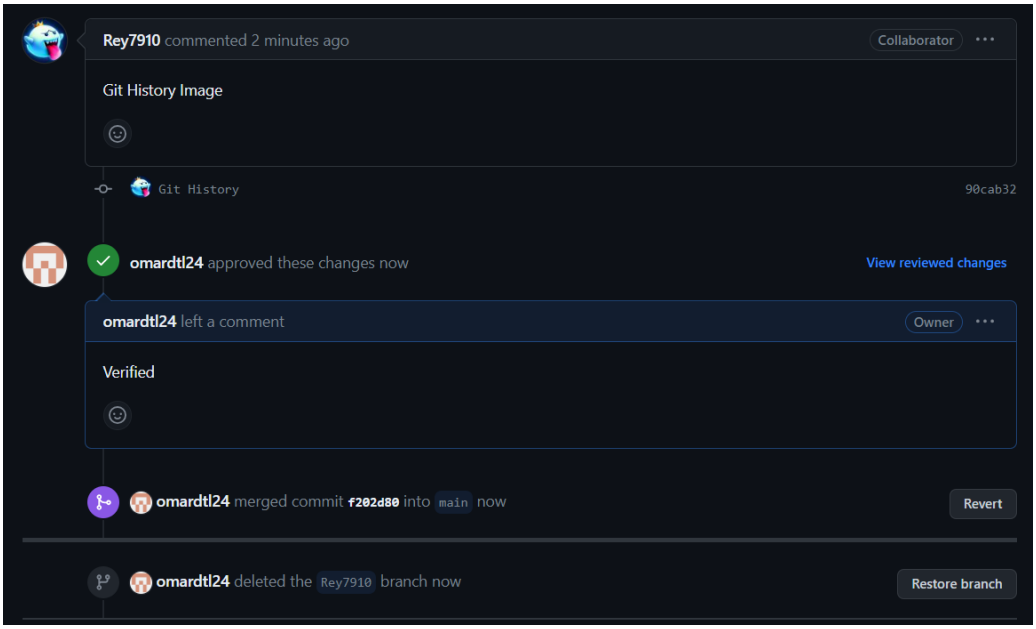
3. Crear una carpeta llamada "*Additional Files*" e incluir una de las imágenes adjuntas en el documento (Cada integrante debe elegir una imagen diferente, no importa si no se usan todas las imágenes). Después versiona en la rama trabajada y sube lo realizado en la rama al repositorio remoto.

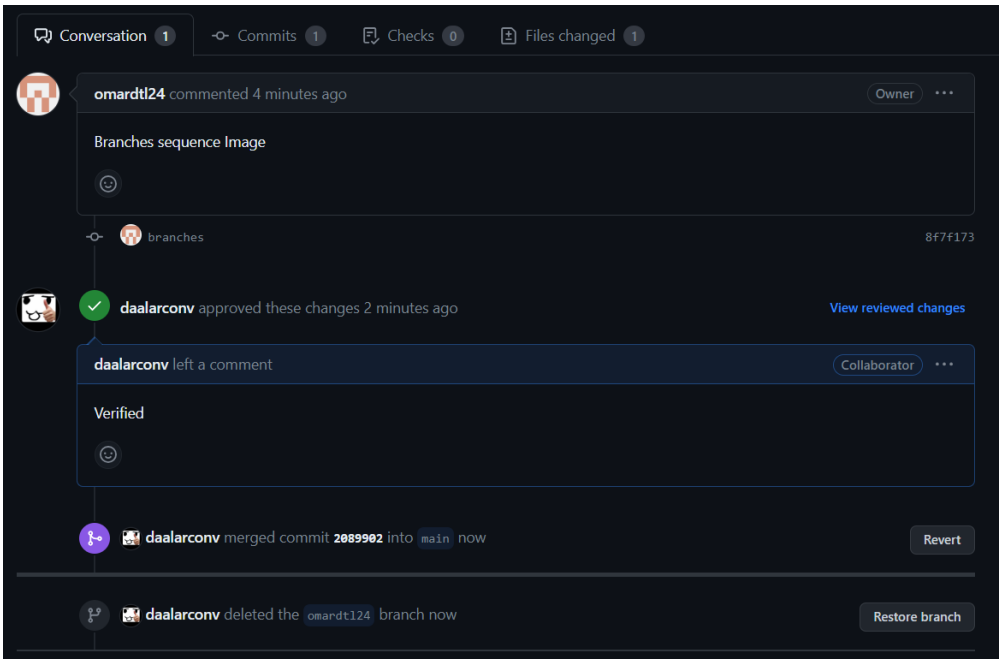


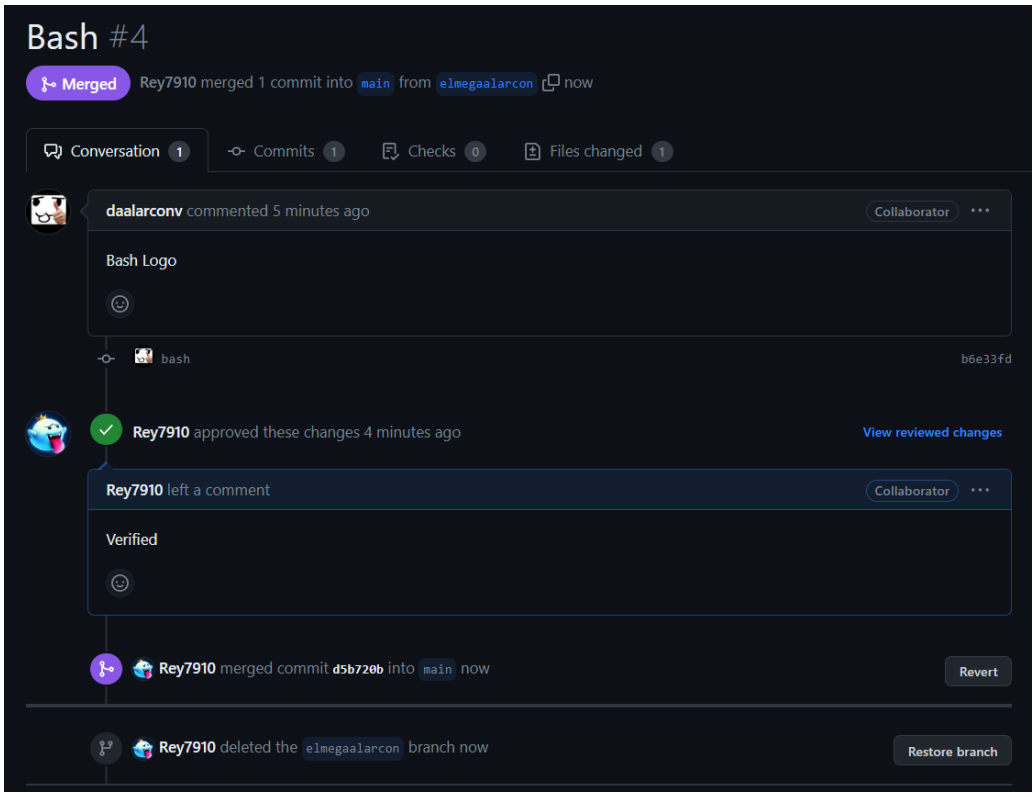

```
omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1
 (omardtl24)
$ git status
On branch omardtl24
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Additional Files/branches.png

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

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1
 (omardtl24)
$ git add .

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1
 (omardtl24)
$ git commit -m "branches"
[omardtl24 8f7f173] branches
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Additional Files/branches.png

omard@LAPTOP-BLK7EOMJ MINGW64 ~/Downloads/Git Lab Files/Git Lab/LabTestIngesoft1
 (omardtl24)
$ git push origin omardtl24
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 16.04 KiB | 16.04 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'omardtl24' on GitHub by visiting:
remote:      https://github.com/omardtl24/LabTestIngesoft1/pull/new/omardtl24
remote:
To https://github.com/omardtl24/LabTestIngesoft1
 * [new branch]      omardtl24 -> omardtl24
```

4. **(Este paso solo se aplica única y exclusivamente cuando todos los integrantes hayan terminado los tres primeros)** Realizar el respectivo Merge a la rama principal en el repositorio remoto; y eliminar la rama auxiliar con la que se trabajó.









A continuación se adjunta el repositorio remoto de ejemplo utilizado en este laboratorio para que sea implementado como guía:

[Ejemplo Laboratorio Gestor de Versiones]

FORMATO DE ENTREGA

Para la entrega de este laboratorio, se requiere únicamente un documento en el que se evidencie con **imágenes y descripciones cortas** el paso a paso realizado en el flujo de trabajo (Sección d del apartado de Desarrollo del laboratorio).

Adicionalmente, se debe incluir un enlace al repositorio el cual debe configurarse como **PÚBLICO**.