

PROGRAMACION I

Trabajo Práctico Nº 2: Git y GitHub

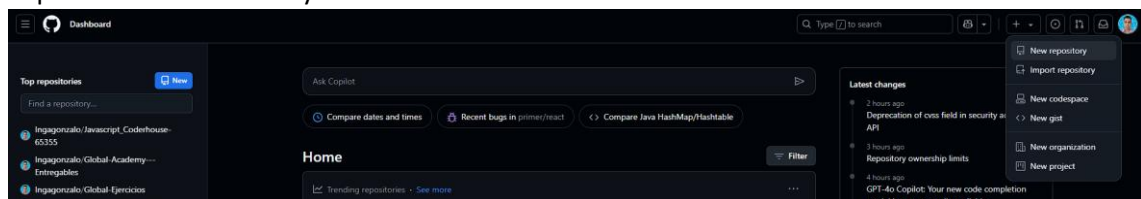
1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

1. ¿Qué es GitHub?

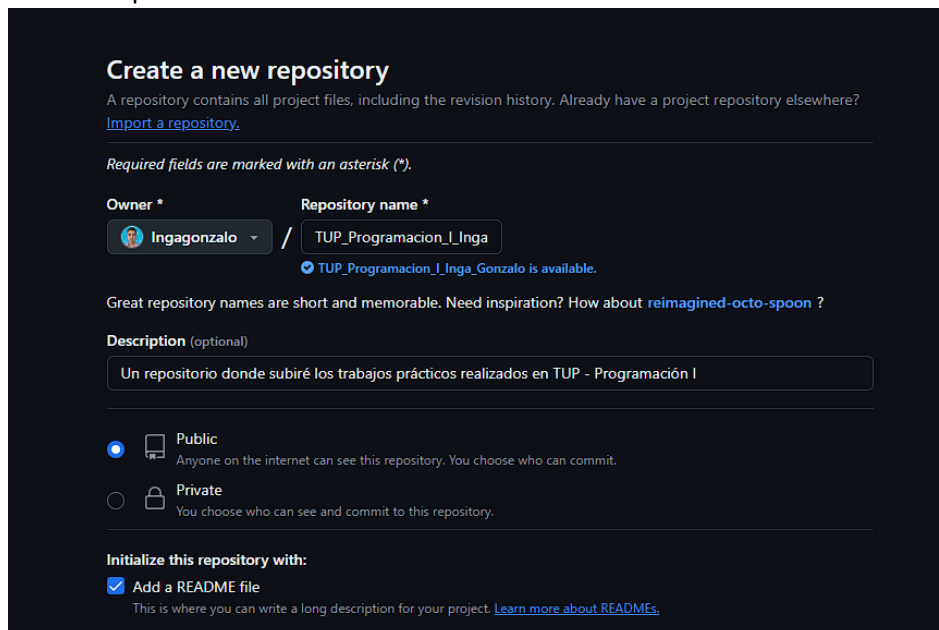
Es una plataforma en la nube que permite a los desarrolladores compartir, guardar proyectos, así como trabajar en conjunto con otros desarrolladores por medio del uso de ramas y versiones, permitiendo trabajar en el mismo código de manera simultánea. GitHub nos permite realizar un seguimiento del proceso de los proyectos, así como también tener un registro de los cambios y saber quiénes los realizaron, permitiendo volver a versiones anteriores en caso de errores en el proyecto.

2. ¿Cómo crear un repositorio en GitHub?

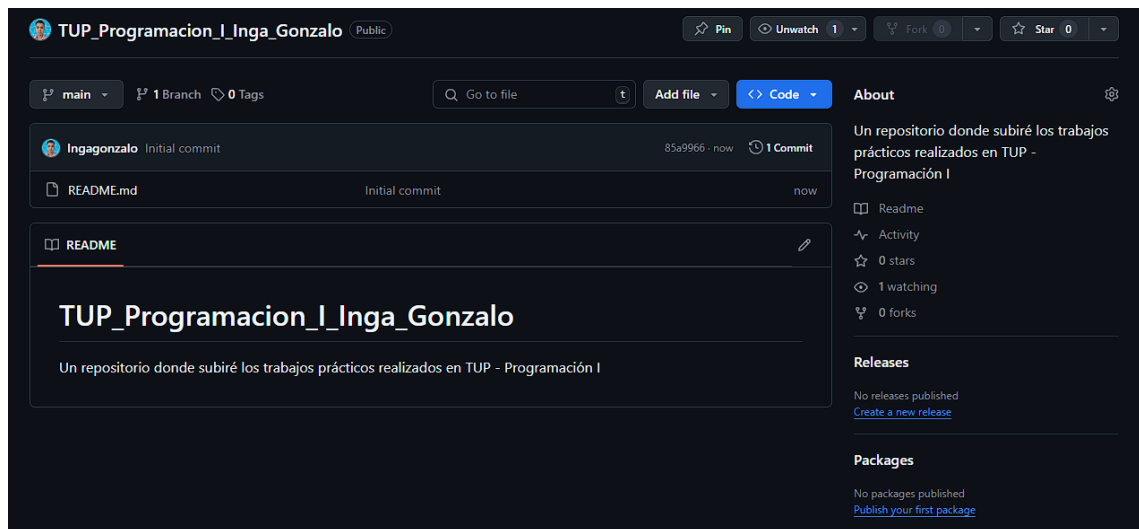
Para empezar necesitamos tener instalado Git en nuestra computadora, luego entramos a la plataforma de GitHub y nos creamos una cuenta.



Creamos un Nuevo repositorio y completamos los datos que GitHub nos solicita y creamos nuestro repositorio.



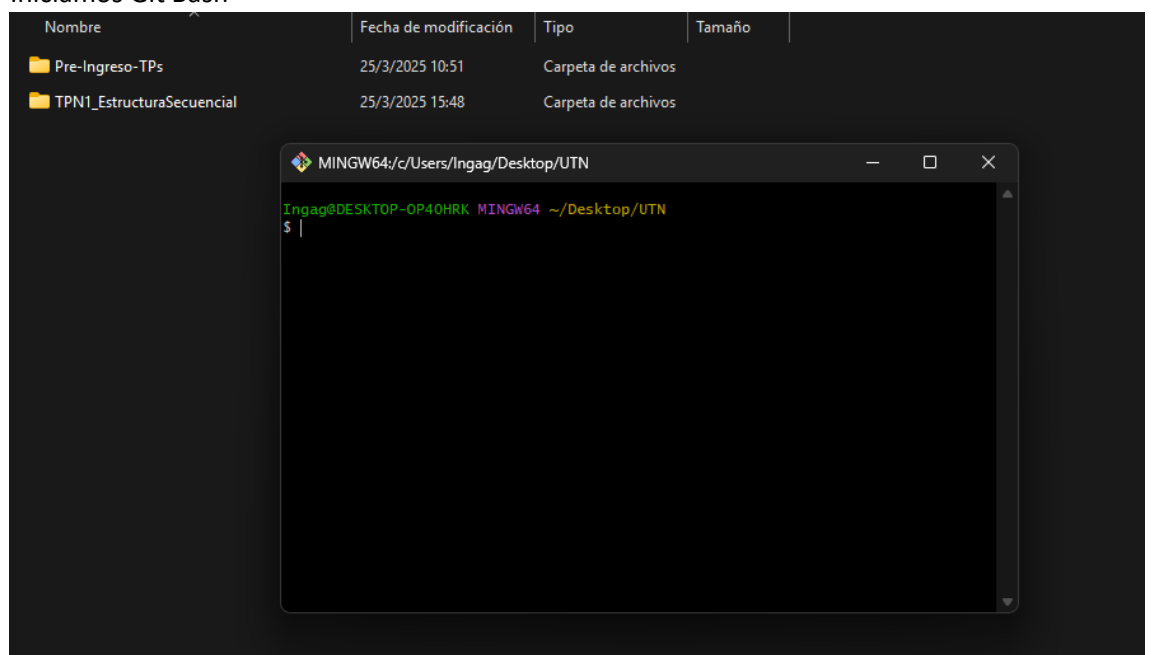
Así se ve cuando creamos nuestro repositorio antes de subir nuestros proyectos.



Ahora nos vamos a nuestra carpeta del primer Trabajo Practico, o de la carpeta o proyecto que deseemos subir a nuestro GitHub.

Y vamos a utilizar los siguientes comandos.

1. Iniciamos Git Bash



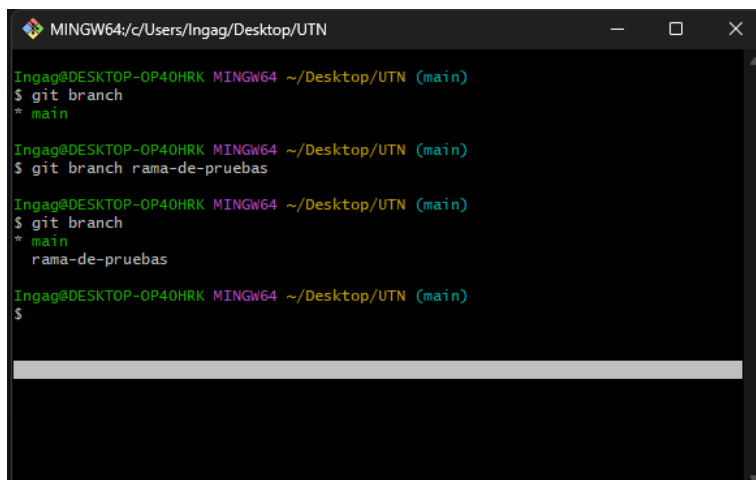
2. Colocamos los siguientes comandos:

- i. Git init (iniciamos Git en nuestra carpeta)
- ii. Git status (vemos el estado de nuestra carpeta y vemos que cambios hubo desde la última vez)
- iii. Git add . (actualizamos todos los cambios)
- iv. Git commit -m "Primer Commit" (realizamos el primer commit con un comentario)
- v. git config --global user.email ingagonzalo1999@gmail.com (configuramos nuestro usuario de git)

- vi. `git config --global user.name Ingagonzalo` (configuramos Nuestro nombre de usuario en git)
- vii. `git branch -M main` (cambiamos de rama a la rama de main)
- viii. `git remote add origin`
https://github.com/Ingagonzalo/TUP_Programacion_I_Inga_Gonzalo.git
(conectamos con nuestro repositorio al que queremos subir nuestros proyectos)
- ix. `git push -u origin main` (subimos nuestro proyecto)

3. ¿Cómo crear una rama en Git?

Para crear una rama en git, en nuestra terminal usando el comando `git branch` podemos verificar que ramas existen en nuestro proyecto y nos indicara en que rama estamos actualmente. Para crear una nueva rama usaremos el comando `git branch nombre-nueva-rama`



```
MINGW64:/c:/Users/Ingag/Desktop/UTN
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$ git branch
* main

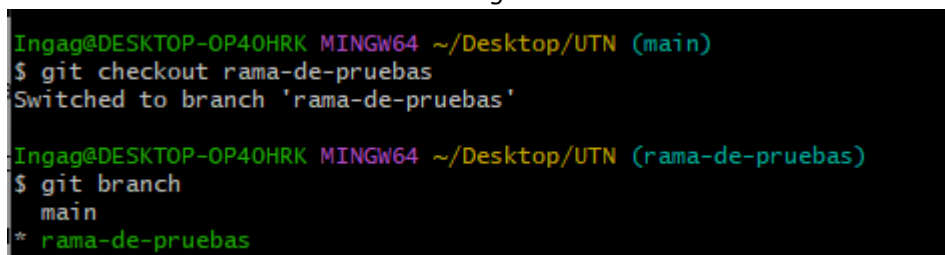
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$ git branch rama-de-pruebas

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$ git branch
* main
  rama-de-pruebas

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$
```

4. ¿Cómo cambiar a una rama en Git?

Para cambiar debemos usar el comando `git checkout nombre-de-la-rama`.



```
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$ git checkout rama-de-pruebas
Switched to branch 'rama-de-pruebas'

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (rama-de-pruebas)
$ git branch
  main
* rama-de-pruebas
```

5. ¿Cómo fusionar ramas en Git?

En este caso fusionare la rama que cree anteriormente `rama-de-pruebas` con mi `main` que es generalmente nuestra rama principal, para hacer esto nos colocamos en nuestra rama `main` y utilizaremos el comando `git merge nombre-de-la-rama` esto fusionara la rama con nuestro `main`

```
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (rama-de-pruebas)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$ git merge rama-de-pruebas
Already up to date.
```

6. ¿Cómo crear un commit en Git?

Para crear un commit debemos:

- Actualizar los cambios realizados en la rama desde la ultima versión utilizando el comando `git add`. o verificar si hay cambios con `git status` (no podremos realizar un commit de una rama que no tenga cambios)
- Luego utilizamos el comando `git commit -m "mensaje-del-commit"`

7. ¿Cómo enviar un commit a GitHub?

Para enviar un commit realizamos el procedimiento anterior y luego utilizamos el comando `git push` para actualizar los cambios en nuestro repositorio de GitHub o como es nuestro caso de que tenemos una rama nueva, debemos usar el siguiente comando `git push -u origin nombre-de-la-rama`.

8. ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión de un repositorio, almacenado en un servidor. Se utiliza para compartir proyectos con otros desarrolladores y nos sirve para mantener copias de seguridad de nuestro repositorio local.

9. ¿Cómo agregar un repositorio remoto a Git?

Para realizar una copia remota en Git se utiliza el comando `git remote add url`, también podemos agregar un nombre de referencia agregando al comando un nombre antes de la url, `git remote add nombre_remoto url`. Luego podemos verificar si nos vinculamos correctamente con el comando `git remote -v`

```
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (rama-de-pruebas)
$ git remote -v
origin  https://github.com/Ingagonzalo/TUP_Programacion_I_Inga_Gonzalo.git (fet
h)
origin  https://github.com/Ingagonzalo/TUP_Programacion_I_Inga_Gonzalo.git (pus
h)
```

10. ¿Cómo empujar cambios a un repositorio remoto?

Antes de realizar cualquier cambio en un repositorio, debemos revisar que no hay cambios pendientes en mi repositorio local, por lo que tengo que usar el comando `git status`. Agregar los cambios con `git add`. y agregar un commit para guardarlo en el historial de versiones con `git commit -m "Mensaje"`.

Luego utilizaremos el comando `git push -u origin main` para empujar los cambios realizados en mi repositorio local hacia el repositorio remoto.

11. ¿Cómo tirar de cambios de un repositorio remoto?

Para actualizar una rama desde el repositorio remoto se utiliza el comando `git pull origin main`, este comando se utiliza para descargar los cambios desde el main y los fusiona con nuestra rama actual.

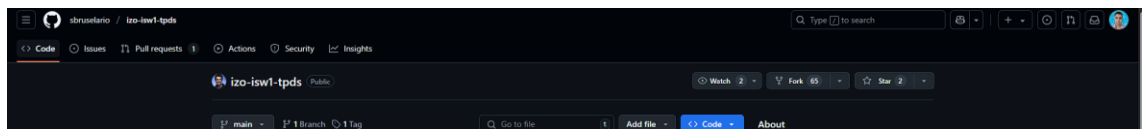
Existe la posibilidad de que los cambios en el repositorio remoto choquen con los del repositorio local, en este caso git solicitara que se resuelvan de manera manual.

12. ¿Qué es un fork de repositorio?

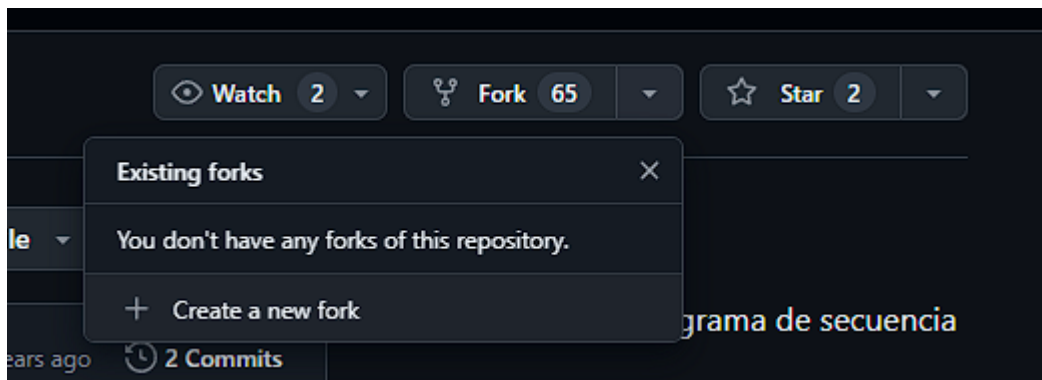
Un fork de un repositorio es una copia en nuestra cuenta de GitHub de algún otro repositorio, es decir un clon de otro repositorio remoto de algún otro usuario, esto nos permite colaborar entre varios desarrolladores sin afectar al proyecto original.

13. ¿Cómo crear un fork de un repositorio?

Para realizar un fork de un repositorio lo que debemos hacer es entrar al repositorio que deseamos clonar en nuestra cuenta de GitHub, y seleccionar **Fork**.



Creamos un nuevo fork.



Editamos lo que querramos cambiar (Nombre del repositorio, descripción y si queremos copiar solo el main o todas las ramas) y creamos el Fork.

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk (*).

Owner *



Ingagonzalo

Repository name *

izo-isw1-tpds

izo-isw1-tpds is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

Trabajo práctico - Diagrama de secuencia

☒ Copy the `main` branch only

Contribute back to sbruselario/izo-isw1-tpds by adding your own branch. [Learn more.](#)

You are creating a fork in your personal account.

Create fork



izo-isw1-tpds

Public

forked from [sbruselario/izo-isw1-tpds](#)

Pin

Watch

0

main



1 Branch



0 Tags

Go to file



Add file



Code

This branch is up to date with [sbruselario/izo-isw1-tpds:main](#).



Contribute



Sync fork



sbruselario TPI 2023

c54caa0 · 2 years ago

2 Commits

diagrama

Primera versión.

3 years ago

readme.md

TPI 2023

2 years ago

README

IZO - Ingeniería de Software 1

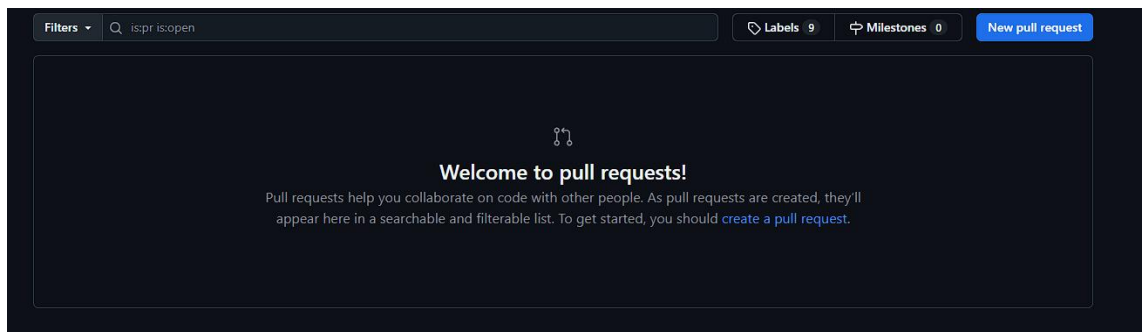
- Identifique los requerimientos.
- Confeccione el diccionario de datos.
- Identifique las reglas de negocio.
- Realice el caso de uso.
- Construya el diagrama entidad-relación y el diagrama de clases.
- Realice el diagrama de secuencia.

14. ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

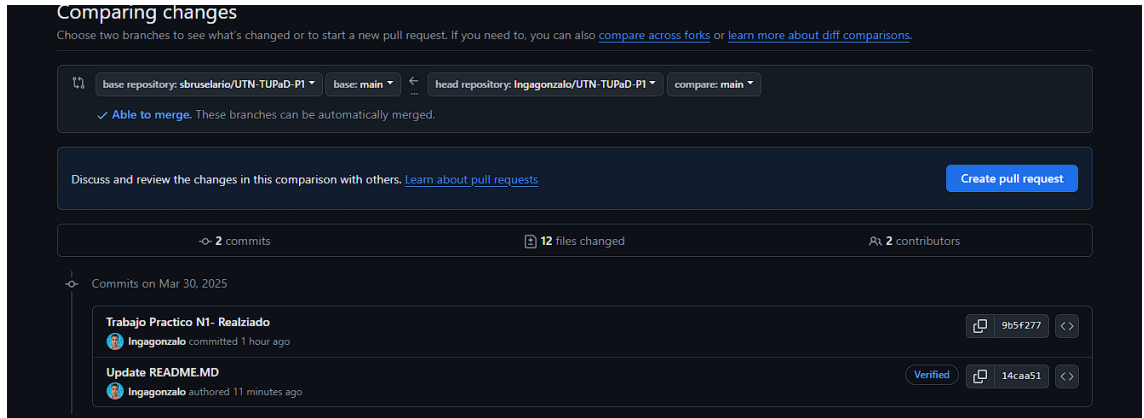
Para realizar un pull request, debemos ir a la sección de **Pull request**.



Seleccionamos **New pull request**



En esta sección podremos ver todos los cambios realizados, es decir, su historial de cambios del repositorio original. Seleccionamos **Create pull request**



En esta sección podemos colocar un título y explicar el porque consideramos que es necesario realizar el **Pull Request**.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#). [Learn more about diff comparisons here](#).

base repository: sbruselario/UTN-TUPaD-P1 base: main head repository: Ingagonzalo/UTN-TUPaD-P1 compare: main

✓ **Able to merge.** These branches can be automatically merged.

Add a title

Trabajo Practico N1 - Realizado

Add a description

Write Preview

Se realizó el trabajo práctico N°1.

Markdown is supported Paste, drop, or click to add files

Allow edits by maintainers

Create pull request

Luego, el pull request se ha realizado, el usuario autor del repositorio recibirá la notificación y decidirá si los cambios realizados aportan al repositorio original.

Trabajo Practico N1 - Realizado #19

Open Ingagonzalo wants to merge 2 commits into sbruselario:main from Ingagonzalo:main

Conversation 0 Commits 2 Checks 0 Files changed 12

Ingagonzalo commented now

Se realizó el trabajo práctico N°1.

Ingagonzalo and others added 2 commits 1 hour ago

- Trabajo Practico N1- Realizado 9b5f277
- Update README.MD 14caa51

Verified

No conflicts with base branch
Changes can be cleanly merged.

15. ¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción lo que debemos hacer es ir a la pestaña de **pull request** y seleccionar la solicitud que se quiere revisar, los cambios aparecerán en la pestaña de **Files Changed**.

En el caso de que los cambios no sean aceptados, se puede dejar comentarios, también en líneas específicas del código, y luego, a través de **Request Changes**, situado en la pestaña **Review Changes**, podemos solicitar revisión.

En el caso de que los cambios sean aceptados debemos ir al botón de **Merge Pull Request**. Donde nos permite varios modos de fusión con nuestro main, siendo por defecto, **Create a merge commit**, que mantiene el historial completo.

16. ¿Qué es una etiqueta en Git?

Git nos da la posibilidad de etiquetar puntos específicos del historial como importantes. Se clasifican en dos tipos:

Annotated Tags (etiqueta anotada): se almacenan en la base de datos como objetos completos, es decir que Git las reconoce como objetos independientes con información adicional (nombre del autor, fecha, etc) Se recomienda para versiones oficiales del software.

Lightweight Tag (etiqueta ligera): no tiene datos adicionales y se usa para referencias temporales que no comprometen el historial del proyecto.

17. ¿Cómo crear una etiqueta en Git?

Para Lightweight tag: `git tag nombre-de-la-etiqueta`

Para Annotated tag: `git tag -a nombre -m "Finalización del módulo de chequeo de datos"` Donde -a indica que es una etiqueta anotada, un nombre representar un acontecimiento importante, -m que hay un mensaje y entre comillas el propósito de la etiqueta.

18. ¿Cómo enviar una etiqueta a GitHub?

Para subir todas las etiquetas que se crearon en el repositorio local ejecutar:

`git push --tags` Para una etiqueta específica se debe ejecutar: `git push origin nombre_tag`

19. ¿Qué es un historial de Git?

Git nos permite tener un registro completo de los cambios realizados en el tiempo, de un proyecto en concreto, esto incluye datos como la fecha y cada commit realizado.

Esto nos ayuda a tener un registro de evolución del proyecto desde su creación.

20. ¿Cómo ver el historial de Git?

Para poder ver el historial de un proyecto debemos usar el comando `git log`.

21. ¿Cómo buscar en el historial de Git?

Al mismo comando `git log` se puede agregar `--author` para autor, `--grep` para buscar textos o palabras o `--since/--until` para fechas, entre otros, para restringir la búsqueda.

22. ¿Cómo borrar el historial de Git?

Para borrar el historial de Git, se debe utilizar el comando `git reset` y existen distintas formas de utilizarlo:

- **git reset:** Quita del stage todos los archivos y carpetas del proyecto.
- **git reset nombreArchivo:** Quita del stage el archivo indicado.

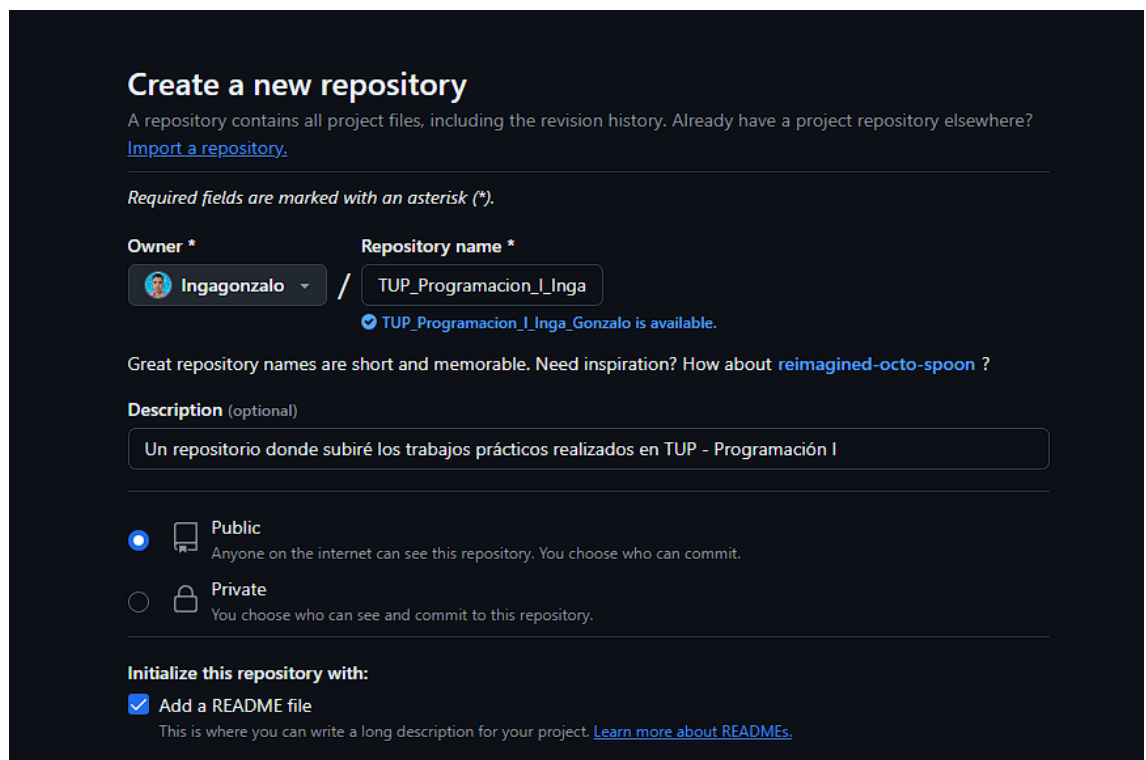
- **git reset nombreCarpeta/**: Quita del stage todos los archivos de esa carpeta.
- **git reset nombreCarpeta/nombreArchivo**: Quita ese archivo del stage (que a la vez está dentro de una carpeta).
- **git reset nombreCarpeta/*.*extensión**: Quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta del stage.

23. ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un repositorio donde el contenido solo puede ser accedido por usuarios autorizados. A diferencia de los repositorios públicos, que cualquier persona puede ver y clonar, los privados restringen el acceso únicamente a los colaboradores autorizados. Esto se utiliza para proyectos en desarrollo o que contienen información privada que no deseas compartir públicamente.

24. ¿Cómo crear un repositorio privado en GitHub?

En el campo de configuración de privacidad al crear un nuevo repositorio, se debe seccionar **Private**



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * Ingagonzalo / **Repository name *** TUP_Programacion_I_Inga
✔ TUP_Programacion_I_Inga_Gonzalo is available.

Great repository names are short and memorable. Need inspiration? How about [reimagined-octo-spoon](#) ?

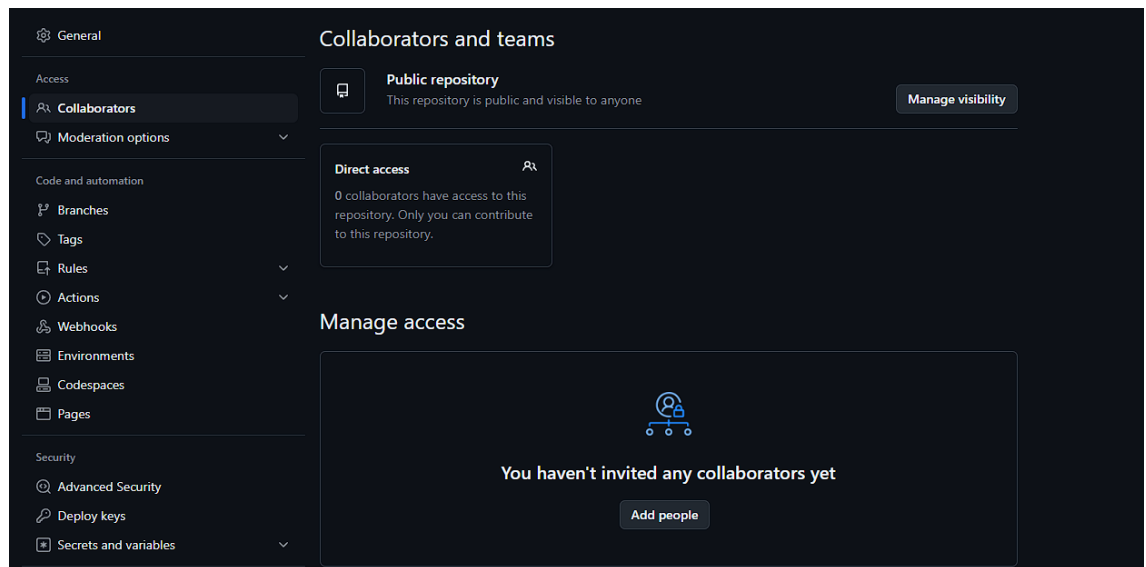
Description (optional)
Un repositorio donde subiré los trabajos prácticos realizados en TUP - Programación I

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

25. ¿Cómo invitar a alguien a un repositorio privado en GitHub?



En la sección de **Collaborators**, para agregar a usuarios de GitHub a tu proyecto, se debe hacer click en **Add people** e ingresar el nombre de usuario de GitHub de tus colaboradores, a su vez se le puede asignar un nivel de acceso de acuerdo a su rol:

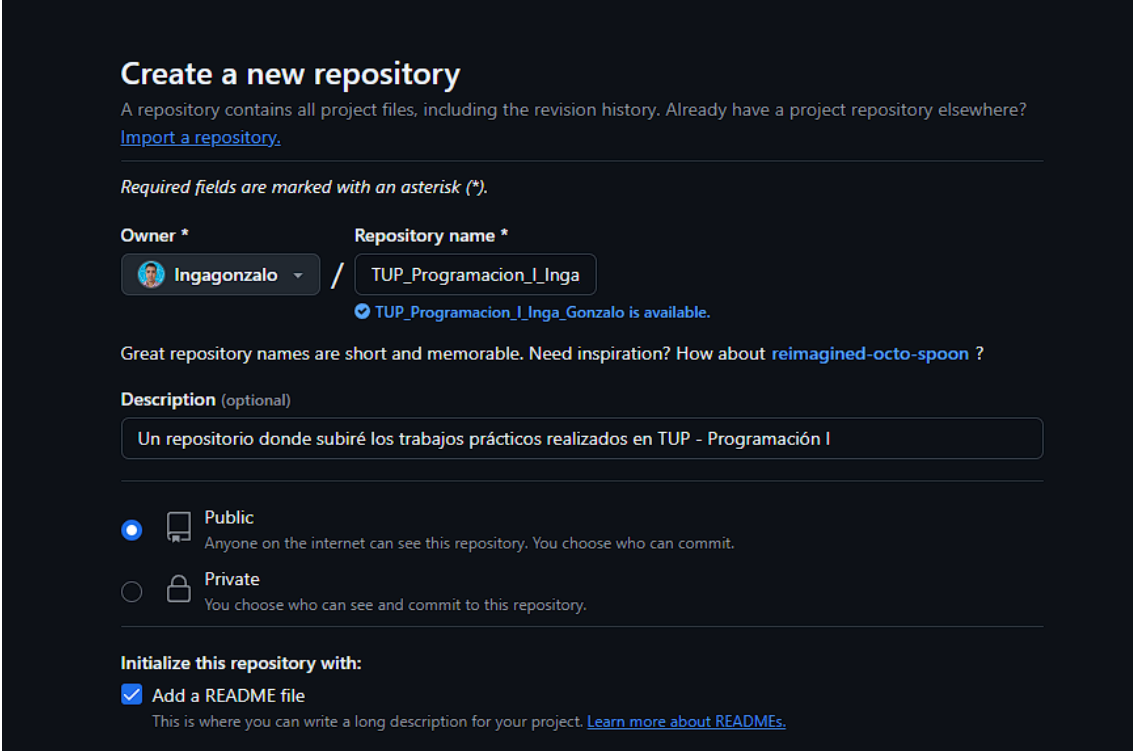
- **Read** sólo para ver el código,
- **Write** para hacer cambios y crear ramas
- **Admin** para gestionar configuraciones y colaboradores.

26. ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es accesible para cualquier persona, permitiendo que cualquiera lo vea y lo clone. Además, son completamente gratuitos. Sin embargo, al igual que los repositorios privados, se requiere permiso para realizar modificaciones en él.

27. ¿Cómo crear un repositorio público en GitHub?

En el campo de configuración de privacidad al crear un nuevo repositorio, se debe seccionar **Public**



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*


Owner * Ingagonzalo / **Repository name *** TUP_Programacion_I_Inga


✔ TUP_Programacion_I_Inga_Gonzalo is available.

Great repository names are short and memorable. Need inspiration? How about [reimagined-octo-spoon](#) ?

Description (optional)

Un repositorio donde subiré los trabajos prácticos realizados en TUP - Programación I

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

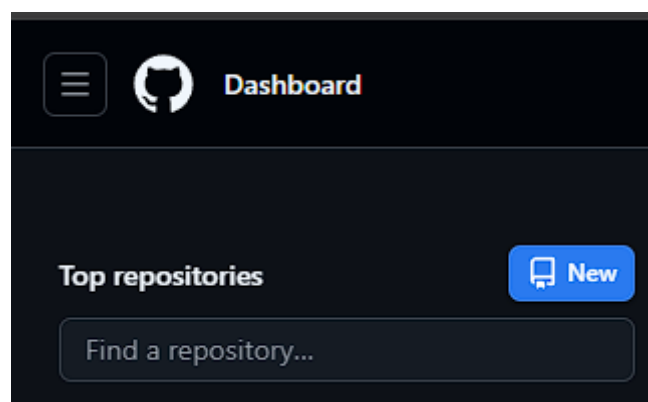
☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

28. ¿Cómo compartir un repositorio público en GitHub?

La manera mas sencilla de compartir un repositorio publico es compartir el enlace directo (la URL).

2) Realizar la siguiente actividad:

- Crear un repositorio.
 - o Dale un nombre al repositorio.
 - o Elije el repositorio sea público.
 - o Inicializa el repositorio con un archivo.




Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 Ingagonzalo ▾

Repository name *

/ TUP_UTN-1er_Cuatrimestr

✔ TUP_UTN-1er_Cuatrimestre is available.

Great repository names are short and memorable. Need inspiration? How about [congenial-waffle](#) ?

Description (optional)

Un repositorio donde subiré los trabajos prácticos realizados en TUP_UTN del primer cuatrimestre

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: **None** ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **None** ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

- Agregando un Archivo
 - o Crea un archivo simple, por ejemplo, "mi-archivo.txt".
 - o Realiza los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"` en la línea de comandos.
 - o Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).

```
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$ git add .

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$ git commit -m "Agregando las Carpetas de las Materias"
[main 575e294] Agregando las Carpetas de las Materias
15 files changed, 1 insertion(+), 100 deletions(-)
create mode 100644 AySo/TP1-AySO-Inga_Gonzalo.docx
create mode 100644 AySo/TP2-AySO-Inga_Gonzalo.docx
create mode 100644 AySo/TP2-AySO-Inga_Gonzalo.pdf
create mode 160000 Programacion/UTN-TUPaD-P1
delete mode 100644 TPN1_EstructuraSecuencial/10_promedio.py
delete mode 100644 TPN1_EstructuraSecuencial/1_holaMundo.py
delete mode 100644 TPN1_EstructuraSecuencial/2_saludo.py
delete mode 100644 TPN1_EstructuraSecuencial/3_datosUsuario.py
delete mode 100644 TPN1_EstructuraSecuencial/4_circulo.py
delete mode 100644 TPN1_EstructuraSecuencial/5_horas.py
delete mode 100644 TPN1_EstructuraSecuencial/6_tablaDeMultiplicar.py
delete mode 100644 TPN1_EstructuraSecuencial/7_calculadoraSimple.py
delete mode 100644 TPN1_EstructuraSecuencial/8_IMC.py
delete mode 100644 TPN1_EstructuraSecuencial/9_conversorTemperatura.py
delete mode 100644 TPN1_EstructuraSecuencial/true.py

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$ git branch -m main
bash: git: command not found

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$ git branch -m main

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$ git remote add origin https://github.com/Ingagonzalo/TUP_UTN-1er_Cuatrimestre.git

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$ git push -u origin main
Enumerating objects: 42, done.
Counting objects: 100% (42/42), done.
Delta compression using up to 16 threads
Compressing objects: 100% (37/37), done.
Writing objects: 100% (42/42), 2.08 MiB | 1.78 MiB/s, done.
Total 42 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Ingagonzalo/TUP_UTN-1er_Cuatrimestre.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$
```


- Creando Branchs
 - o Crear una Branch
 - o Realizar cambios o agregar un archivo
 - o Subir la Branch

```
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (main)
$ git checkout -b nueva-rama
Switched to a new branch 'nueva-rama'

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (nueva-rama)
$ git branch
  main
* nueva-rama
  rama-de-pruebas

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (nueva-rama)
$ git status
On branch nueva-rama
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Matematicas/
    Organizacion Empresarial/

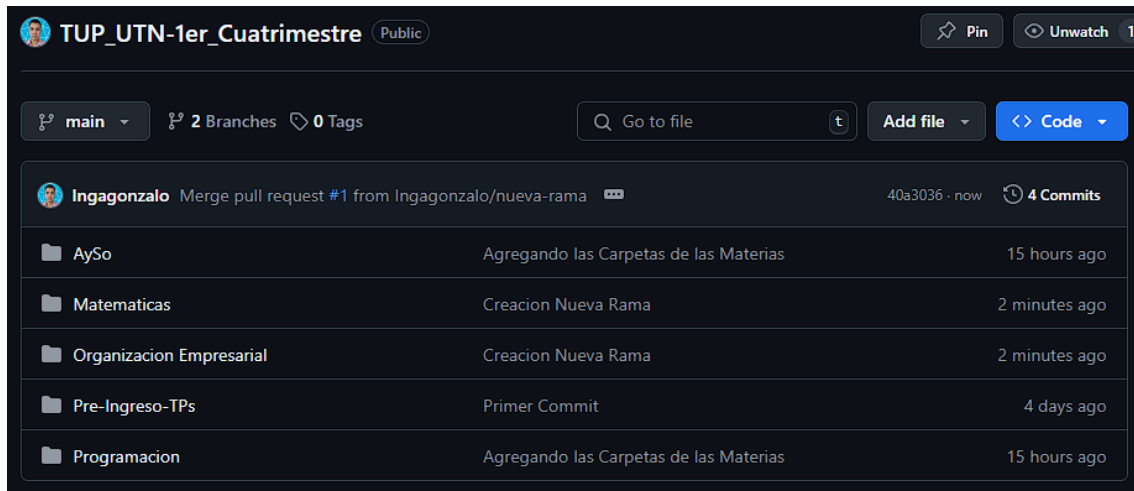
nothing added to commit but untracked files present (use "git add" to track)

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (nueva-rama)
$ git add .

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (nueva-rama)
$ git commit -m "Creacion Nueva Rama"
[nueva-rama 1c96c5c] Creacion Nueva Rama
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Matematicas/Matematicas.txt
 create mode 100644 Organizacion Empresarial/Organizacion_empresarial.txt

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (nueva-rama)
$ git push origin nueva-rama
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 518 bytes | 518.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'nueva-rama' on GitHub by visiting:
remote:   https://github.com/Ingagonzalo/TUP_UTN-1er_Cuatrimestre/pull/new/nueva-rama
remote:
To https://github.com/Ingagonzalo/TUP_UTN-1er_Cuatrimestre.git
 * [new branch]   nueva-rama -> nueva-rama

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/UTN (nueva-rama)
$ |
```

3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub


- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz clic en "Create repository".

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*


Owner * **Repository name ***


 Ingagonzalo / conflict-exercise

✔ conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about [symmetrical-succotash](#) ?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: **None**


Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

[Create repository](#)

Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio (usualmente algo `https://github.com/tuusuario/conflict-exercise.git`).
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando: `git clone https://github.com/tuusuario/conflict-exercise.git`
- Entra en el directorio del repositorio: `cd conflict-exercise`

```
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise
$ git clone https://github.com/Ingagonzalo/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:
`git checkout -b feature-branch`
- Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo: Este es un cambio en la feature branch.
- Guarda los cambios y haz un commit:
`git add README.md`
`git commit -m "Added a line in feature-branch"`

```
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise
$ cd conflict-exercise

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (feature-branch)
$ git add README.md

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 9d3aea0] Added a line in feature-branch
1 file changed, 2 insertions(+), 1 deletion(-)

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (feature-branch)
$
```

Paso 4: Volver a la rama principal y editar el mismo archivo

- Cambia de vuelta a la rama principal (main):
`git checkout main`
- Edita el archivo README.md de nuevo, añadiendo una línea diferente: Este es un cambio en la main branch.
- Guarda los cambios y haz un commit:
`git add README.md`
`git commit -m "Added a line in main branch"`

```
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (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")

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (main)
$ git add .

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 0b3043a] Added a line in main branch
1 file changed, 2 insertions(+), 1 deletion(-)
```

Paso 5: Hacer un merge y generar un conflicto

- Intenta hacer un merge de la feature-branch en la rama main:
git merge feature-branch
- Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md.

```
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (main)
$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

Paso 6: Resolver el conflicto

- Abre el archivo README.md en tu editor de texto. Verás algo similar a esto:
<<<<<<< HEAD
Este es un cambio en la main branch.
=====
Este es un cambio en la feature branch.
>>>>>>> feature-branch
- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
- Edita el archivo para resolver el conflicto y guarda los cambios (Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).

- Añade el archivo resuelto y completa el merge:
`git add README.md`
`git commit -m "Resolved merge conflict"`

```
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (main|MERGING)
$ git add .

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main 87a87f4] Resolved merge conflict

Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (main)
$ git merge feature-branch
Already up to date.

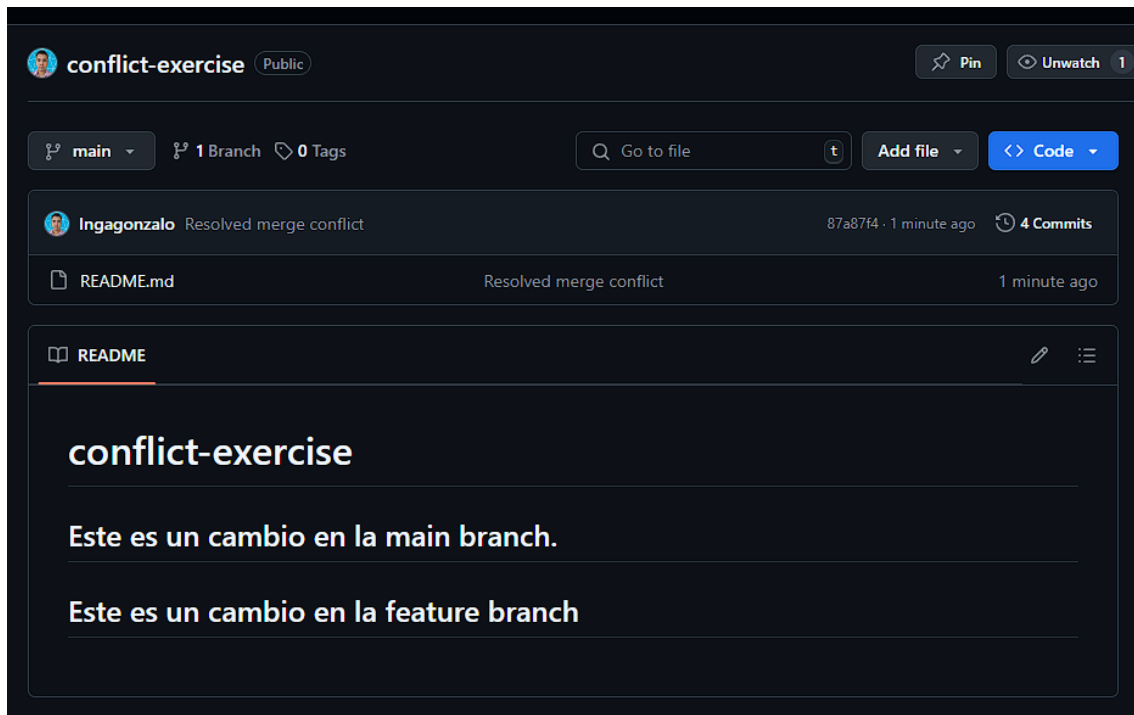
Ingag@DESKTOP-OP40HRK MINGW64 ~/Desktop/conflict-exercise/conflict-exercise (main)
$ |
```

Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub:
`git push origin main`
- También sube la feature-branch si deseas:
`git push origin feature-branch`

Paso 8: Verificar en GitHub

- Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.
- Puedes revisar el historial de commits para ver el conflicto y su resolución.



URL: <https://github.com/Ingagonzalo/conflict-exercise>