

ALUMNA: ARJONA, PAOLA YANINA

Comisión 24

Programación 1

Trabajo Práctico N°2: Git y GitHub

Objetivo

El estudiante desarrollará competencias para trabajar con Git y GitHub, aplicando conceptos fundamentales de control de versiones, colaboración en proyectos y resolución de conflictos, en un entorno simulado y guiado.

Resultados de aprendizaje:

1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.
2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto.
3. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo.
4. Resolver conflictos en un entorno de control de versiones: Identificar, analizar y solucionar conflictos de merge generados en un flujo de trabajo con múltiples ramas.

Desarrollo de Actividades

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

a) ¿Qué es GitHub?

Es una plataforma en línea que permite almacenar proyectos que usan el sistema de control de versiones Git. Es el servicio más popular del mundo para compartir y colaborar en proyectos de software, tanto de código abierto como privados.

b) ¿Cómo crear un repositorio en GitHub?

Paso 1: Ingresar a <https://github.com> e iniciar sesión

Paso 2: Click en new

Paso 3: Configurar el nuevo repositorio (nombre, descripción, etc.)

Paso 4: Click en create repository.

Nota: para clonar el proyecto, hacer click en "code" y copiar la dirección.

En la carpeta elegida abrir la terminal, colocar el siguiente comando:

`git clone https://github.com/tuusuario/conflict-exercise.git`

Luego entrar en el directorio del repositorio usando comando: `cd "nombre-directorio"`

c) ¿Cómo crear una rama en Git?

Para crear una nueva rama, comando: **`git branch <nueva-rama>`**

Para verificar las ramas disponibles, comando: **`git branch`**

d) ¿Cómo cambiar a una rama en Git?

Para cambiar a una rama, comando: **git checkout <rama>**

Para crear y cambiar a nueva rama: **git checkout -b <nombre-rama>**

e) ¿Cómo fusionar ramas en Git?

Para fusionar una rama con la actual, comando: **git merge <nueva-rama>**

f) ¿Cómo crear un commit en Git?

Agregar los cambios, mediante comando: **git add .** (agrega todos los archivos al Stage) y **git add arch1 arch2 ... archN** (agrega archivos específicos)

Luego se puede crear el commit con un mensaje descriptivo, comando: **git commit -m "mensaje"**

Guarda los archivos preparados como una nueva versión del proyecto.

g) ¿Cómo enviar un commit a GitHub?

Abrir la terminal o consola.

Ir a la carpeta del proyecto

Escribir **git add <archivo>**

Escribir **git commit -m "mensaje"** para crear el commit.

Escribir **git push origin main** para enviar el commit a GitHub.

h) ¿Qué es un repositorio remoto?

Es una copia de un proyecto, se aloja en un servidor remoto (plataformas como GitHub o GitLab). Permite sincronizar el trabajo con otros desarrolladores.

i) ¿Cómo agregar un repositorio remoto a Git?

Para añadir un nuevo repositorio remoto, comando: **git remote add origin <url>**

j) ¿Cómo empujar cambios a un repositorio remoto?

Para subir cambios al repositorio remoto, comando: **git push -u origin master**

git push (luego de la primera vez)

*Nota: **origin** es el alias que por defecto se le suele dar al repositorio remoto.*

Nota: La rama principal se puede llamar master o main.

k) ¿Cómo tirar de cambios de un repositorio remoto?

Para aplicar cambios desde repositorio remoto, comando: **git pull origin master**

git pull (Si usamos **-u** en el push)

Descarga los cambios desde el repositorio remoto y los fusiona con tu versión local.

l) ¿Qué es un fork de repositorio?

Un fork es una copia completa de un repositorio de GitHub que se crea dentro de tu cuenta.

m) ¿Cómo crear un fork de un repositorio?

- ✓ Ingresar a un repositorio público de GitHub (por ejemplo, <https://github.com/otro-usuario/proyecto>).
- ✓ Hacer clic en el botón Fork (esquina superior derecha).
- ✓ Elegir tu cuenta para crear una copia del proyecto en tu propio GitHub.
- ✓ Configurar lo deseado (nombre, descripción, etc.)
- ✓ Click en create fork

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

Pull requests: Para proponer cambios a proyectos. Permiten revisiones, comentarios y aprobaciones antes de fusionar el código.

- ✓ Ir a GitHub e ingresar al repositorio
- ✓ Hacer click en "Compare & pull request".
- ✓ Click en "New pull request"
- ✓ Click en "Create pull request".

o) ¿Cómo aceptar una solicitud de extracción?

- ✓ Ir a la pestaña "Pull Requests" del repositorio.
- ✓ Click en la solicitud pendiente.
- ✓ Revisar los archivos y comentarios.
- ✓ Click en "Merge pull request".
- ✓ Confirmar con "Confirm merge".

p) ¿Qué es una etiqueta en Git?

Una etiqueta (tag) es un marcador que señala un punto específico en la historia de Git, normalmente usado para marcar versiones (por ejemplo: v1.0, v2.1.3).

q) ¿Cómo crear una etiqueta en Git?

Existen dos tipos de etiquetas: Etiqueta ligera (lightweight tag) y Etiqueta anotada (annotated tag)

Para crear tag, comando: **git tag nombre**

r) ¿Cómo enviar una etiqueta a GitHub?

Para subir cambios incluyendo todos los tags, comando: **git push --tags**

s) ¿Qué es un historial de Git?

Es el registro de todos los cambios realizados en un repositorio de Git, los cambios se guardan como commit con información sobre quién, cuándo y qué se modificó.

t) ¿Cómo ver el historial de Git?

Ver commits, comando: **git log**

Ver commits (una línea c/u), comando: **git log --oneline**

git log se listan las confirmaciones realizadas en ese repositorio en orden cronológico inverso; es decir, las confirmaciones más recientes aparecen primero.

u) ¿Cómo buscar en el historial de Git?

Para buscar por palabra clave en mensajes de commits: **git log --grep="palabra clave"**

Para buscar cambios en un archivo específico: **git log nombre-del-archivo**

Para ver commits (graficado), comandos: git log --decorate --all --graph --oneline

Para commits hechos por un autor específico: git log --author="Nombre del Autor"

¿Cómo borrar el historial de Git?

Se usa los siguientes comandos:

git reset --modo HEAD^: Volver a commit anterior

git reset --modo HEAD^N: Volver hacia el N° anterior commit

git reset --modo hash-commit: Volver hacia commit específico

git revert hash-commit: Volver hacia commit especificado y generar uno nuevo con su estado

git reflog: Mostrar historial del repositorio

v) ¿Qué es un repositorio privado en GitHub?

Un repositorio privado es un proyecto que solo tú y las personas que invites pueden ver o modificar.

w) ¿Cómo crear un repositorio privado en GitHub?

- ✓ Ir a <https://github.com> e iniciar sesión
- ✓ Click en "New"
- ✓ Escribir el nombre del repositorio.
- ✓ Marcar la opción "Private".
- ✓ Click en "Create repository".

x) ¿Cómo invitar a alguien a un repositorio privado en GitHub?

- ✓ Ir al repositorio
- ✓ Click en la pestaña "Settings"
- ✓ Ve a "Collaborators"
- ✓ Ingresar el nombre de usuario o email
- ✓ Haz clic en "Add people"

y) ¿Qué es un repositorio público en GitHub?

Un repositorio público es aquel que cualquiera puede ver, clonar o descargar, pero solo tú o tus colaboradores pueden modificarlo.

z) ¿Cómo crear un repositorio público en GitHub?

- ✓ Ir a <https://github.com> e iniciar sesión
- ✓ Click en "New"
- ✓ Escribir el nombre del repositorio.
- ✓ Marcar la opción "Public".
- ✓ Click en "Create repository".

aa) ¿Cómo compartir un repositorio público en GitHub?

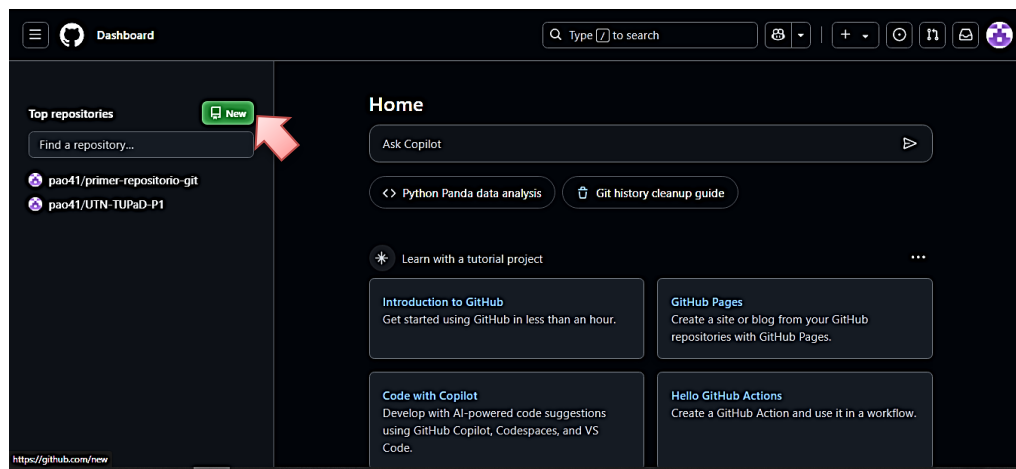
- ✓ Ingresa a github
- ✓ Ir al repositorio público en GitHub.
- ✓ Copiar la URL del navegador o haz clic en el botón "Code" para copiar el enlace.
- ✓ Compartir el enlace

2) Realizar la siguiente actividad:

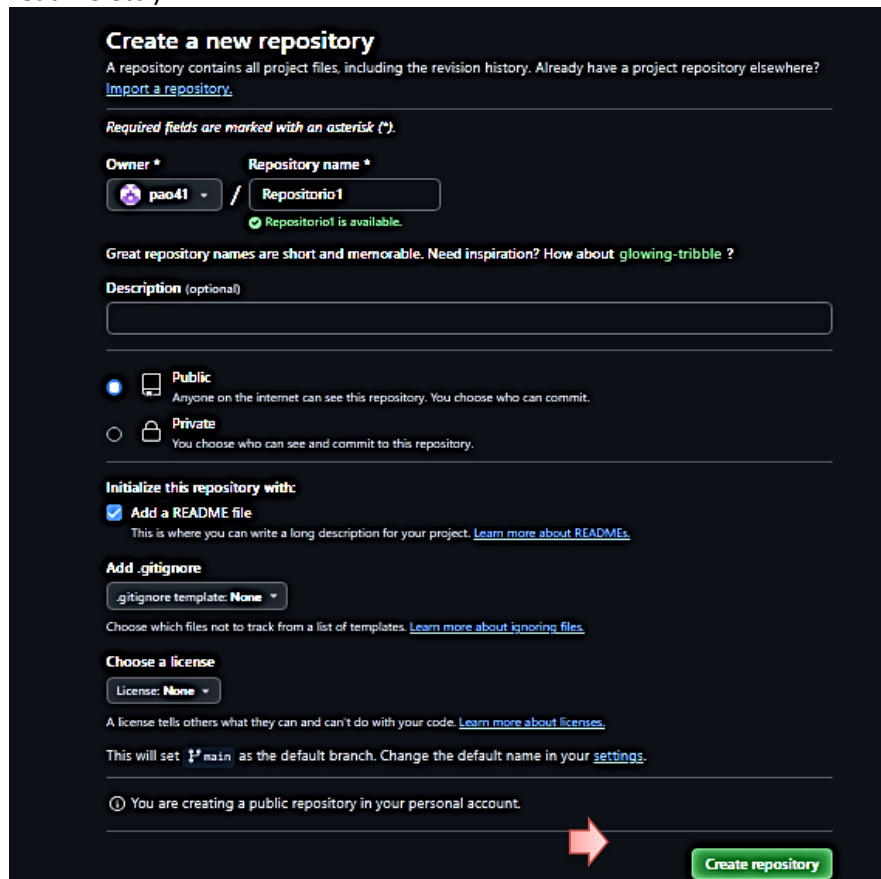
- Crear un repositorio.
 - Dale un nombre al repositorio.
 - Elije el repositorio sea público.
 - Inicializa el repositorio con un archivo.

Paso 1: Ingresar a <https://github.com> e iniciar sesión

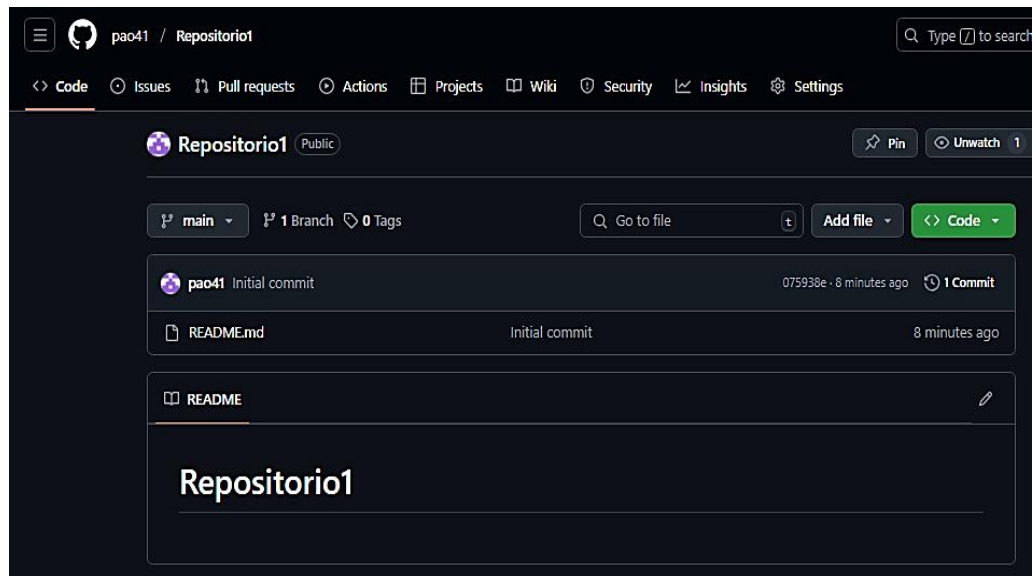
Paso 2: En sección home, hacer click en new



Paso 3: Configurar el nuevo repositorio (nombre, descripción, tildar public, tildar readme etc.)



Paso 4: Click en create repository.



Clonar el repositorio en una carpeta

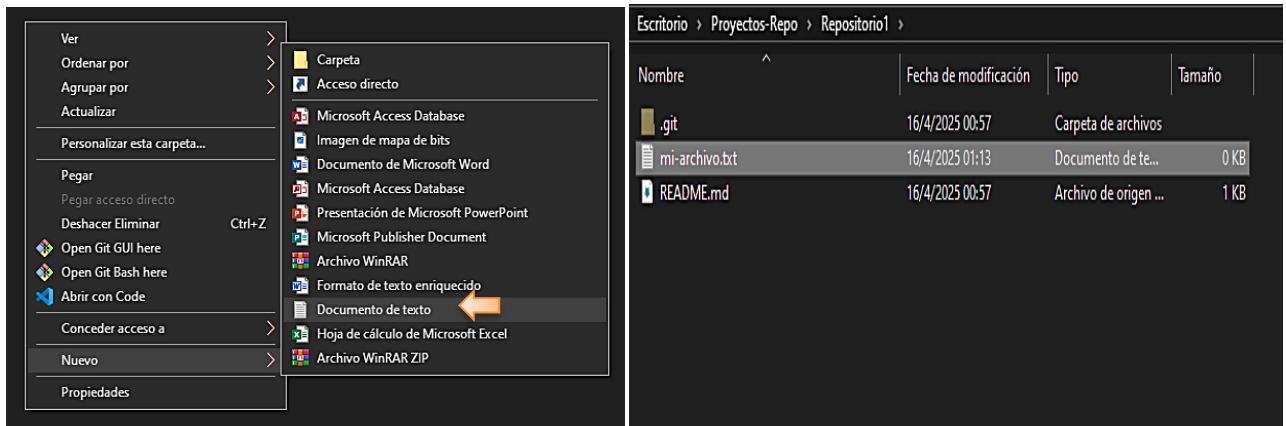
```
MINGW64/c/Users/pao5/Desktop/Proyectos-Repo/Repositorio1
pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo
$ git clone https://github.com/pao41/Repositorio1.git
Cloning into 'Repositorio1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo
$ cd Repositorio1

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (main)
$ |
```

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

En la carpeta, crear un archivo de bloc de notas:



Comandos git add . y git commit -m "Agregando mi-archivo.txt":

```
MINGW64:/c/Users/pao5/Desktop/Proyectos-Repo/Repositorio1
pao5@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (main)
$ git add .

pao5@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (main)
$ git commit -m "Agregando mi-archivo.txt"
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

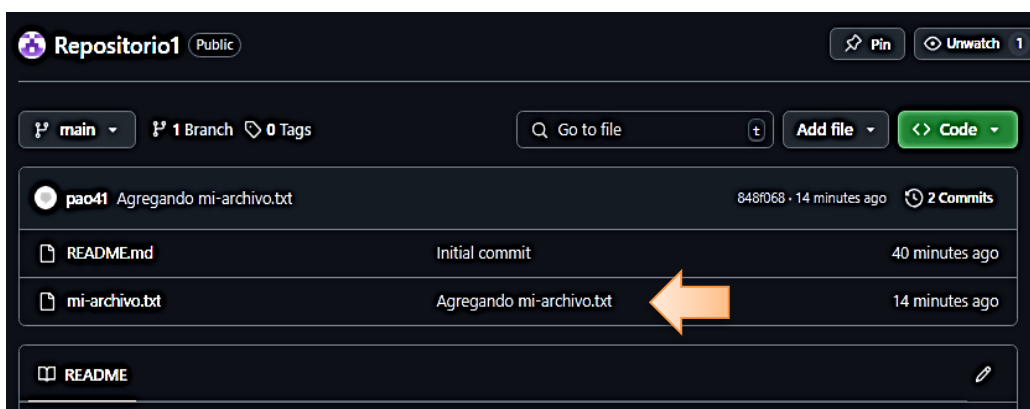
nothing to commit, working tree clean
```

Comando git push origin main:

```
pao5@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes | 47.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/pao41/Repositorio1.git
075938e..848f068 main -> main

pao5@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (main)
$ |
```

Cambios en Github:



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

Listar, crear y cambiar a nueva Branch:

```
MINGW64:/c/Users/pao5/Desktop/Proyectos-Repo/Repositorio1

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (main)
$ git branch
* main

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (main)
$ git branch rama-nueva

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (main)
$ git checkout rama-nueva
M      mi-archivo.txt
Switched to branch 'rama-nueva'

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (rama-nueva)
$ git branch
main
* rama-nueva

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (rama-nueva)
$
```

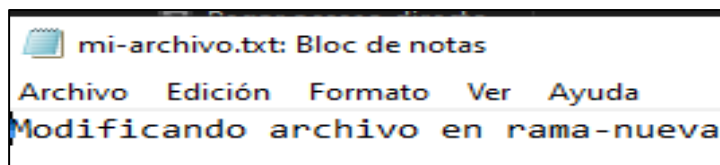
Realizar cambios en archivo:

```
MINGW64:/c/Users/pao5/Desktop/Proyectos-Repo/Repositorio1

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (rama-nueva)
$ echo "Modificando archivo en rama-nueva" > mi-archivo.txt

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (rama-nueva)
$ git add .
warning: in the working copy of 'mi-archivo.txt', LF will be replaced by CRLF the ne
xt time Git touches it

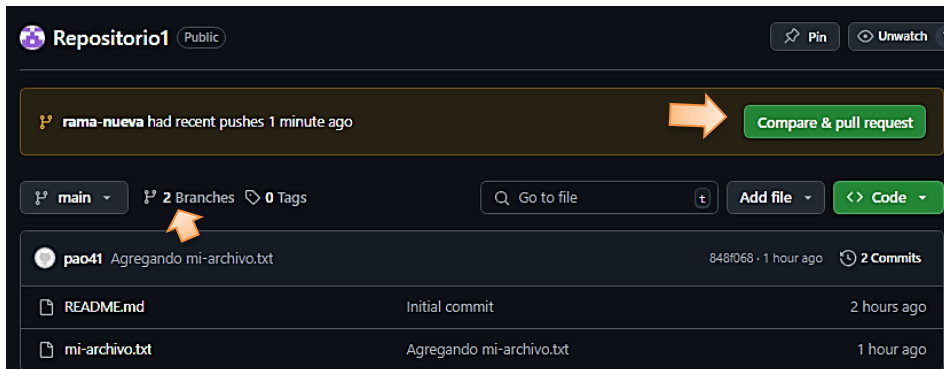
pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (rama-nueva)
$ git commit -m "Realizando modificacion del archivo en rama-nueva"
[rama-nueva adcd05f] Realizando modificacion del archivo en rama-nueva
1 file changed, 1 insertion(+)
```



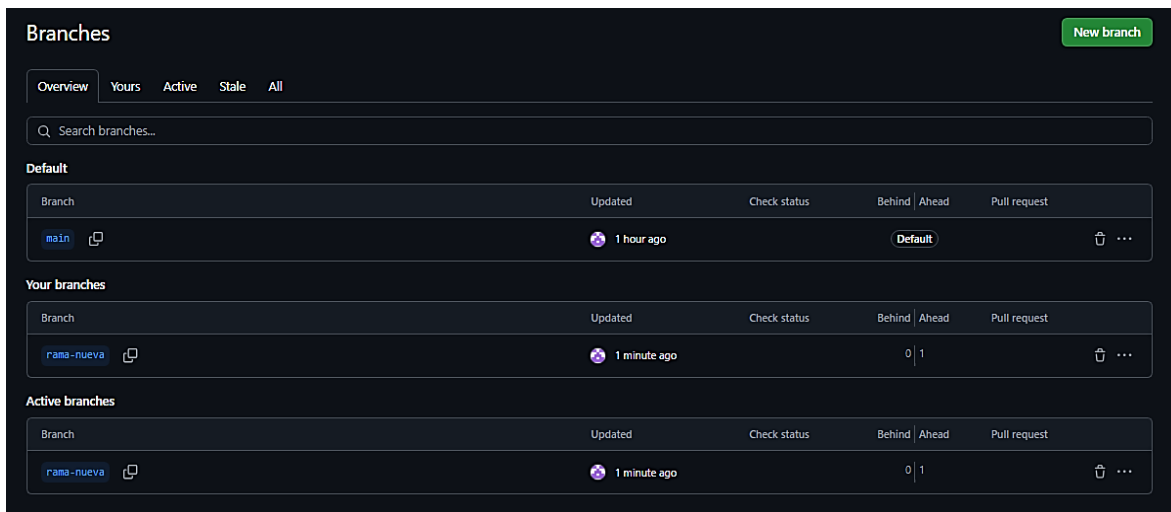
Subir archivo:

```
pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/Repositorio1 (rama-nueva)
$ git push origin rama-nueva
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 335 bytes | 167.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'rama-nueva' on GitHub by visiting:
remote:   https://github.com/pao41/Repositorio1/pull/new/rama-nueva
remote:
To https://github.com/pao41/Repositorio1.git
 * [new branch]      rama-nueva -> rama-nueva
```

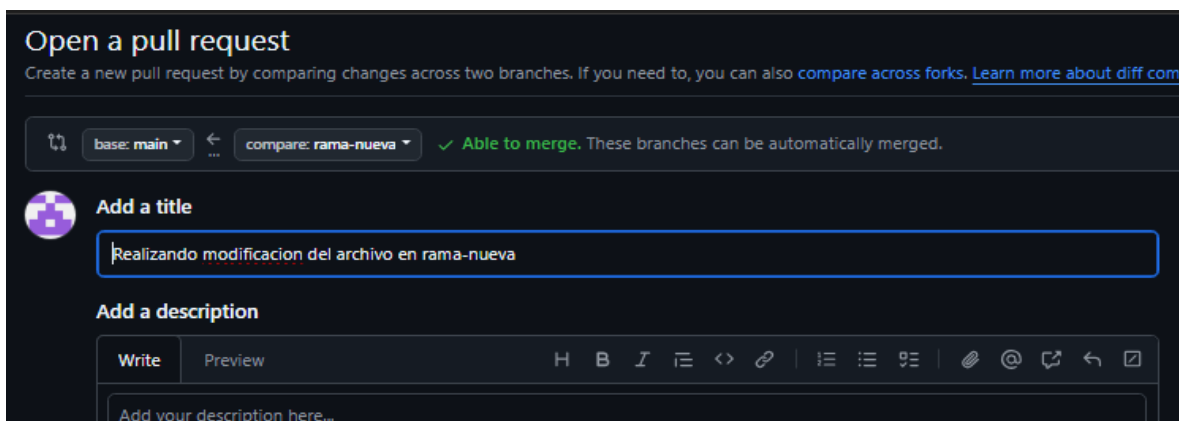
Actualización en github, 2 branches:



Click en “Branches”:



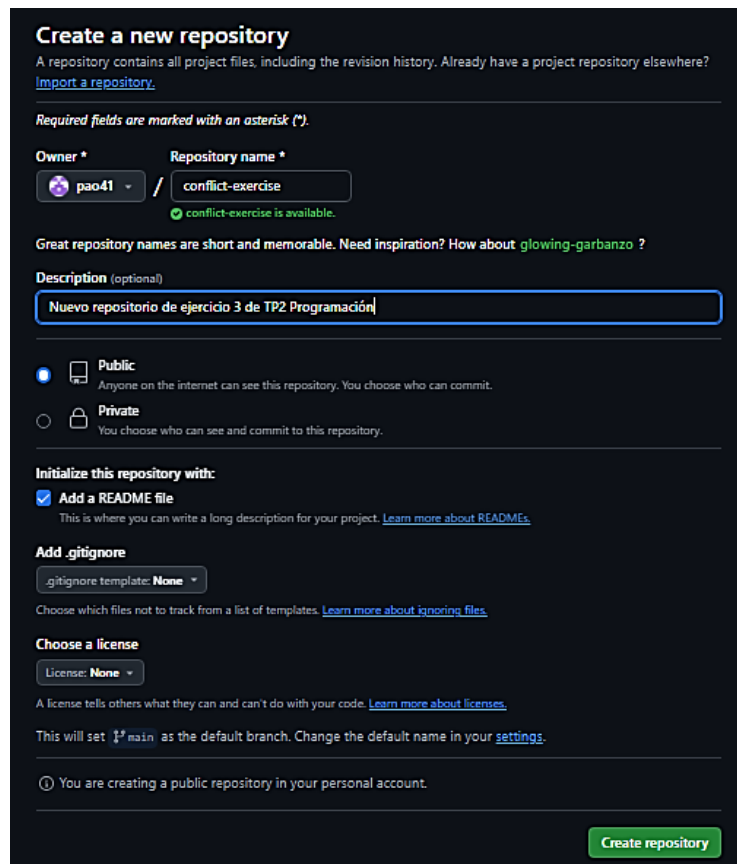
Click en “Compare & pull request”:



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 *

☒ conflict-exercise is available.

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

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:

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

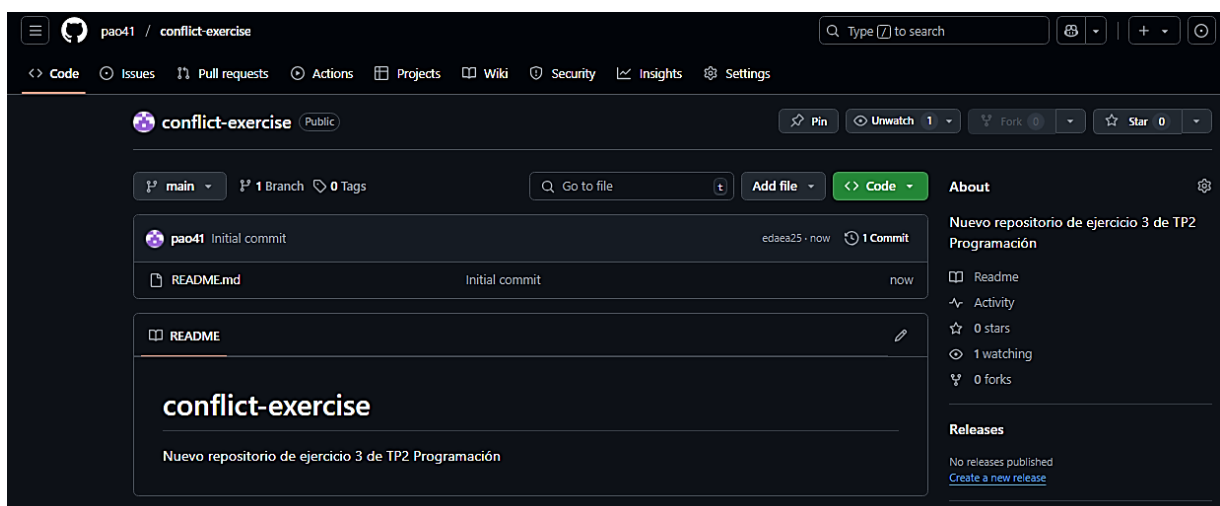
Choose a license
License:

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 como `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`

→ <https://github.com/pao41/conflict-exercise>

```
MINGW64/c/Users/pao5/Desktop/Proyectos-Repo/conflict-exercise
pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo
$ git clone https://github.com/pao41/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo
$ cd conflict-exercise

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/conflict-exercise (main)
$ |
```

Paso 3: Crear una nueva rama y editar un archivo

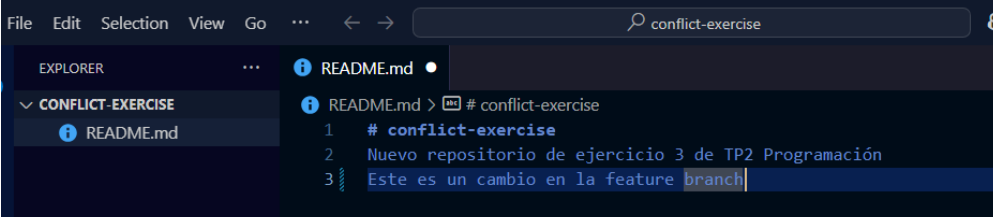
- Crea una nueva rama llamada feature-branch:

`git checkout -b feature-branch`

```
pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/conflict-exercise (feature-b
branch)
$ git branch
* feature-branch
  main
```

- 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*.



```
File Edit Selection View Go ... < > conflict-exercise
EXPLORER
CONFLICT-EXERCISE
  README.md
README.md
1 # conflict-exercise
2 Nuevo repositorio de ejercicio 3 de TP2 Programación
3 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"
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + - [ ] [ ] ... ^

PS C:\Users\pao5_\Desktop\Proyectos-Repo\conflict-exercise> git add README.md
PS C:\Users\pao5_\Desktop\Proyectos-Repo\conflict-exercise> git commit -m "Added a line in feature-branch"
[feature-branch 584ab6b] Added a line in feature-branch
1 file changed, 1 insertion(+)
```

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

- Cambia de vuelta a la rama principal (main): `git checkout main`

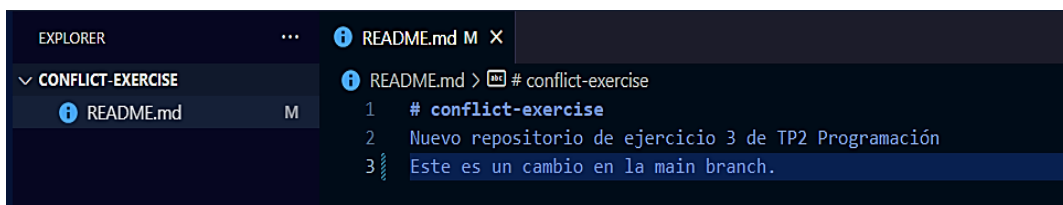


```
MINGW64:/c/Users/pao5_/Desktop/Proyectos-Repo/conflict-exercise

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/conflict-exercise (main)
$
```

- Edita el archivo README.md de nuevo, añadiendo una línea diferente: Este es un cambio en la main branch.



```
EXPLORER ... README.md M X

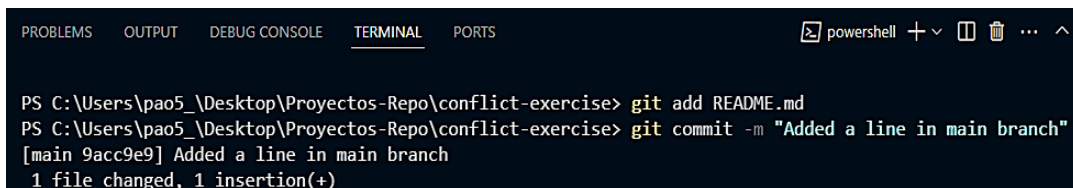
CONFLICT-EXERCISE
  README.md M

README.md > # conflict-exercise
1 # conflict-exercise
2 Nuevo repositorio de ejercicio 3 de TP2 Programación
3 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"
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + - [ ] [ ] ... ^

PS C:\Users\pao5_\Desktop\Proyectos-Repo\conflict-exercise> git add README.md
PS C:\Users\pao5_\Desktop\Proyectos-Repo\conflict-exercise> git commit -m "Added a line in main branch"
[main 9acc9e9] Added a line in main branch
1 file changed, 1 insertion(+)
```

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.

```
MINGW64:/c/Users/pao5/Desktop/Proyectos-Repo/conflict-exercise

pao5@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/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:

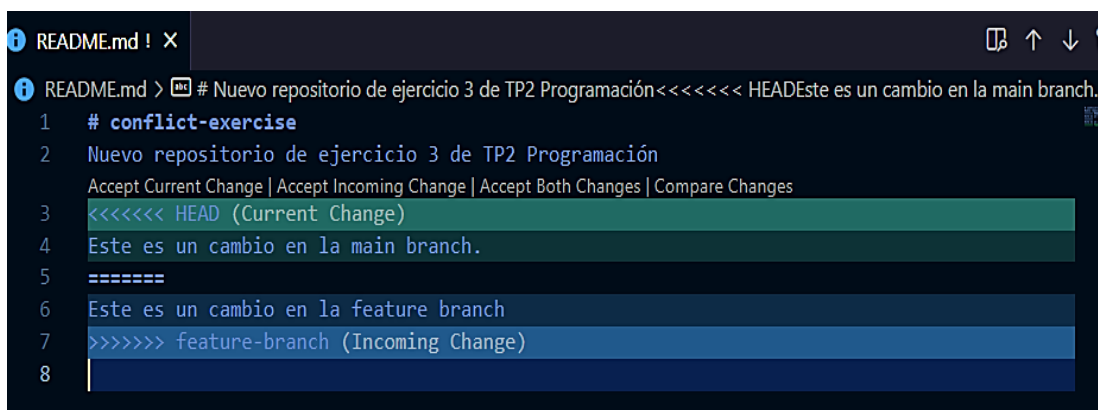
```
<<<<<<< HEAD
```

Este es un cambio en la main branch.

```
=====
```

Este es un cambio en la feature branch.

```
>>>>>>> feature-branch
```



```
README.md ! X
README.md > # Nuevo repositorio de ejercicio 3 de TP2 Programación<<<<<< HEADEste es un cambio en la main branch.
1 # conflict-exercise
2 Nuevo repositorio de ejercicio 3 de TP2 Programación
3 Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
4 <<<<<< HEAD (Current Change)
5 Este es un cambio en la main branch.
6 =====
7 Este es un cambio en la feature branch
8 >>>>>> feature-branch (Incoming Change)
```

- 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).


```
README.md M X
README.md > # conflict-exercise
1 # conflict-exercise
2 Nuevo repositorio de ejercicio 3 de TP2 Programación
3 Este es un cambio en la main branch.
4 Este es un cambio en la feature branch
```

- Añade el archivo resuelto y completa el merge:

```
git add README.md
```

```
git commit -m "Resolved merge conflict"
```

```
MINGW64:/c:/Users/pao5_/Desktop/Proyectos-Repo/conflict-exercise
pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/conflict-exercise (main)
$ git add README.md

pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/conflict-exercise (main)
$ git commit -m "Resolved merge conflict"
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub:

```
git push origin main
```

```
pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/conflict-exercise (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 818 bytes | 272.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/pao41/conflict-exercise.git
   edaea25..3bf8c19  main -> main
```

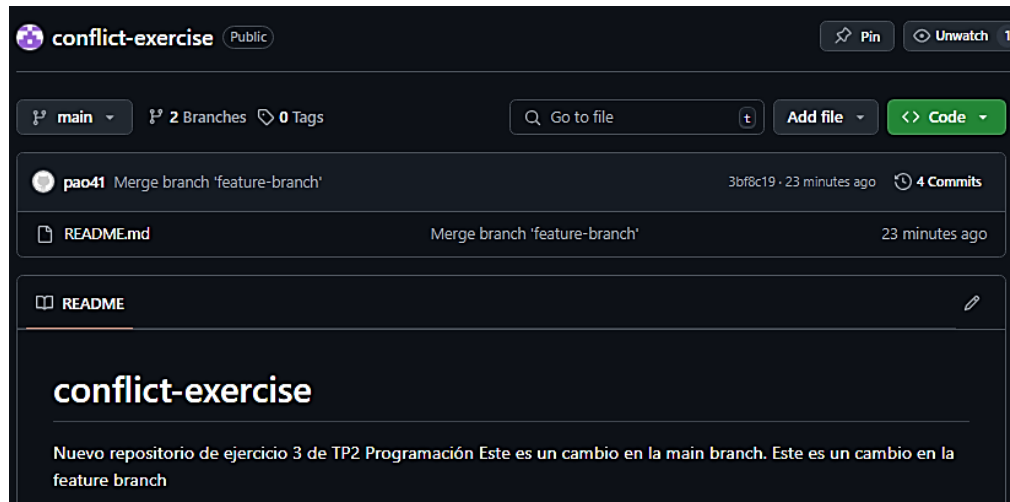
- También sube la feature-branch si deseas:

```
git push origin feature-branch
```

```
pao5_@DESKTOP-973240M MINGW64 ~/Desktop/Proyectos-Repo/conflict-exercise (main)
$ git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/pao41/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/pao41/conflict-exercise.git
 * [new branch]   feature-branch -> 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.

