

PROGRAMACIÓN I

Trabajo Práctico N.º 2: Git y Github

Estudiante: Emilia Gómez Juárez

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.

Actividades:

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

- **¿Qué es GitHub?**

GitHub es la principal plataforma web para la creación de trabajos colaborativos y se basa en el sistema de control de versiones Git. Permite a los desarrolladores crear y compartir repositorios de forma privada o pública, almacenar y gestionar proyectos de software propios y de manera colaborativa.

- **¿Cómo crear un repositorio en GitHub?**

PROGRAMACIÓN I

Para usar GitHub primero debemos crearnos una cuenta, podemos hacerlo desde la web o en GitHub Desktop. Una vez registrados, al iniciar sesión podemos configurar nuestro perfil y comenzar a crear repositorios.

Para crear un repositorio, nos dirigimos a la página principal en la web y arriba a la derecha encontraremos un botón llamado “New”, al clickearlo nos redirige a una página para configurar el nuevo repositorio. Allí elegimos el perfil que será *Owner* del proyecto, el nombre y una descripción del mismo. Podemos también elegir que sea público o privado y agregar o no un README.

Al terminar estas configuraciones, al final de la página encontramos un botón “Create repository”. Al clickear dicho botón, se facilitan los comandos que debemos utilizar para crear el repositorio local y subirlo al repositorio remoto, o en caso de ya existir el repositorio local, los comandos para subir el repositorio local a GitHub.

- **¿Cómo crear una rama en Git?**

Al iniciar un repositorio en Git, automáticamente se crea una rama por defecto llamada en general Master o Main.

Para crear una rama en Git debemos utilizar el comando *git branch*:

```
$ git branch version1
```

- **¿Cómo cambiar a una rama en Git?**

Para cambiar de una rama a otra debemos utilizar el comando *git checkout*:

```
$ git checkout version1
```

- **¿Cómo fusionar ramas en Git?**

Para fusionar ramas en Git debemos utilizar el comando *git merge*. Primero nos ubicaremos en la rama en la cual queremos incorporar los cambios de otra rama (*git checkout rama-destino*). Una vez en la rama de destino, ejecutamos *git merge rama-origen*. Si no hay conflictos Git crea un *commit* de fusión automáticamente, si hay conflictos Git muestra los

PROGRAMACIÓN I

archivos con diferencias que deben ser resueltas. Una vez resuelto el conflicto se realiza un *commit* de la versión corregida y las ramas quedan fusionadas.

- **¿Cómo crear un commit en Git?**

Para hacer un commit primero debo guardar en el repositorio el archivo con los cambios realizados con el comando *git add {nombre del archivo}*. Luego creo el commit utilizando el comando *git commit -m "{mensaje o nombre del commit}"*

- **¿Cómo enviar un commit a GitHub?**

Para enviar un commit a GitHub se utiliza el comando *git push -u origin rama-a-subir*

- **¿Qué es un repositorio remoto?**

Un repositorio remoto es una versión de, por ejemplo, un proyecto de software almacenada en un servidor en la nube.

- **¿Cómo agregar un repositorio remoto a Git?**

Para agregar un repositorio remoto a Git se clona ese repositorio en la computadora, es decir, se descarga una copia del repositorio remoto en la computadora. Para ello se utiliza el comando *git clone URL-del-repositorio*.

- **¿Cómo empujar cambios a un repositorio remoto?**

Para empujar (push) cambios a un repositorio remoto en principio debe haberse hecho al menos un commit. Dada esa condición, se utiliza el comando *git push -u origin rama-a-empujar*

- **¿Cómo tirar de cambios de un repositorio remoto?**

Tirar de cambios (pull) significa que se descargan los cambios más recientes del repositorio remoto, actualizando el repositorio local. Para realizarlo se utiliza el comando *git pull origin nombre-de-la-rama*. Este comando descarga los cambios y los fusiona automáticamente. Otra opción es utilizar el comando *git fetch* que solo descarga los cambios pero no los fusiona, lo que permite revisar los cambios antes de mergearlos.

PROGRAMACIÓN I

- **¿Qué es un fork de repositorio?**

Es una forma de crear una copia de un repositorio que se crea en la cuenta de GitHub. A diferencia de la clonación, el *fork* crea una copia independiente del repositorio sin vinculación con el repositorio original. Permite contribuir a proyectos de código abierto sin modificar el repositorio original, hacer pruebas, crear una versión propia de algún proyecto que ya existe.

- **¿Cómo crear un fork de un repositorio?**

Para crear un fork nos dirigimos a la página en GitHub del repositorio original que se desea copiar, clickeamos el botón *fork* y elegimos la cuenta donde guardar la copia. GitHub crea el fork en el perfil.

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

Luego de subir los cambios a GitHub, vamos al repositorio en la web y nos dirigimos a la pestaña pull request. Allí seleccionamos la rama, escribimos la descripción y creamos el pull request.

- **¿Cómo aceptar una solicitud de extracción?**

Se puede aceptar desde la pestaña “Pull requests”. Es recomendable revisar los cambios realizados antes de aceptar la solicitud. Al clickear la solicitud, GitHub muestra los archivos modificados y los commits. Otra opción es probar los cambios antes de aceptarlos con el comando *git fetch* copiando el proyecto en el local. Si todo está bien, se hace clic en el botón *Merge Pull Request*.

- **¿Qué es una etiqueta en Git? ¿Cómo crear una etiqueta en Git?**

Una etiqueta es una referencia a un estado, una versión importante en un repositorio. Para crearla se utiliza el comando *git tag*. Puede ser una etiqueta ligera en la que no se agrega ningún dato extra y funciona como puntero a un commit específico (*git tag nombre-del-tag*); o anotada que contiene información extra como fecha, mensaje y autor (*git tag -a nombre de la tag -m “mensaje”*).

- **¿Cómo enviar una etiqueta a GitHub?**

PROGRAMACIÓN I

Para enviar una sola etiqueta se utiliza el comando: *git push origin nombre-de-la-etiqueta*. Para enviar todas las etiquetas se utiliza el comando: *Git push –tags*. Para eliminar una etiqueta se utiliza el comando: *git push –delete origin nombre-de-la-etiqueta*

- **¿Qué es un historial de Git? ¿Cómo ver el historial de Git? ¿Cómo buscar en el historial de Git? ¿Cómo borrar el historial de Git?**

El historial de git es el registro de todos los commits de un repositorio. Para ver el mismo se utiliza el comando: *git log*

Este puede utilizar flags para indicar diversos parámetros de búsqueda (fecha, rango de fechas, n° ultimos commits, etc).

Buscar n últimos commits: *git log -n {número de commits a mostrar}*

Rango de fechas: *git log -- since="{fecha}" – until= "{fecha}"*

Log específico: *git log –grep= "{cadena de búsqueda}"*

Autor: *git log –author = "{nombre del/a autor/a}"*

Para eliminar todo el historial de git se podría reestablecer el repositorio a su estado original eliminando la carpeta oculta de git ubicada en la carpeta raíz del repositorio o con el comando: *rm -rf .git*

Para eliminar un commit específico se puede utilizar el código: *git rebase -i* o *git rebase -i {has del commit desde el que se quiere modificar o HEAD para utilizar un número específico de commits}*

- **¿Qué es un repositorio privado en GitHub? ¿Cómo crear un repositorio privado en GitHub? ¿Cómo invitar a alguien a un repositorio privado en GitHub?**

Es un repositorio en el cual solo puede ser visto o accedido por quienes tienen el permiso del propietario para hacerlo. El proceso para crear un repositorio privado es el mismo que el que se realiza para crear cualquier repositorio, es parte de la configuración del repositorio seleccionar la opción privado o público en la web o colocar *private: true* o *false* en la API. En caso

PROGRAMACIÓN I

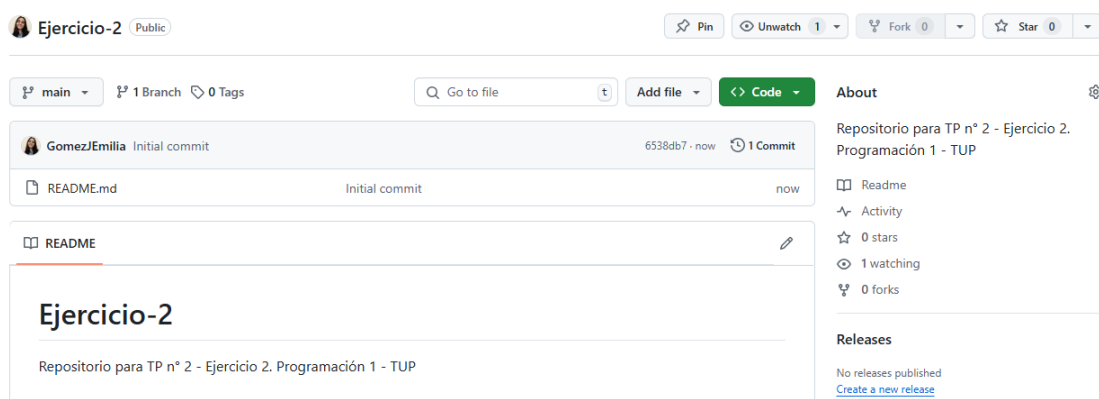
de hacerlo por la CLI de GitHub se debe colocar la flag `--private`, también al momento de crear el repositorio. Para invitar a alguien a un repositorio privado se debe enviar la invitación al mail de dicha persona, para hacerlo debemos dirigirnos al apartado Settings o configuración, manage access y agregar un colaborador.

- **¿Qué es un repositorio público en GitHub? ¿Cómo crear un repositorio público en GitHub? ¿Cómo compartir un repositorio público en GitHub?**

Un repositorio público es aquel que puede ser visto y utilizado por cualquiera que ingrese por el link del mismo o la página del perfil del propietario. Al igual que se dijo en el caso anterior, se debe colocar en el caso de la API `private:false`, en el caso de la web seleccionar la opción Público, y en el caso de la CLI `--public`. Para compartirlo simplemente es necesario enviar el link del repositorio.

2) Realizar la siguiente actividad:

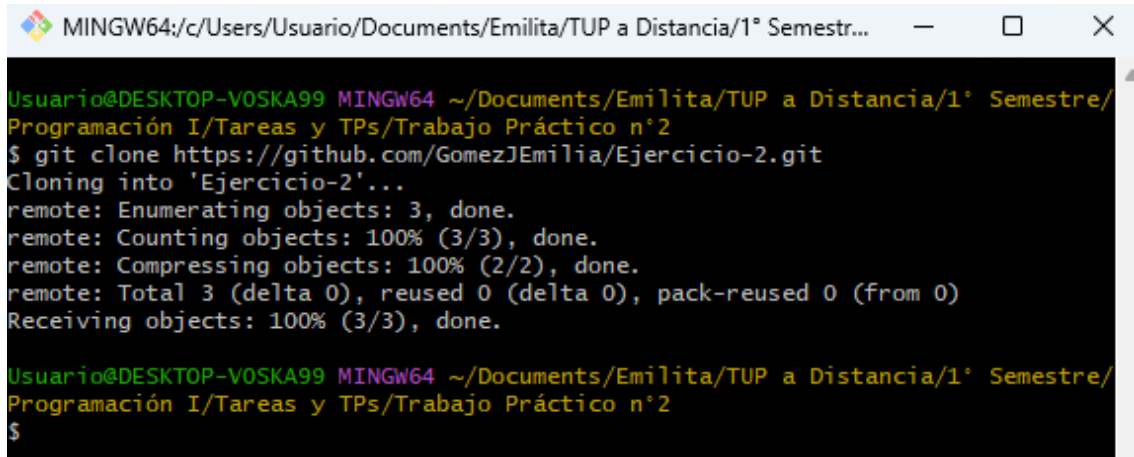
- Crear un repositorio.
 - a. Dale un nombre al repositorio.
 - b. Elije el repositorio sea público.
 - c. Inicializa el repositorio con un archivo.



- Agregando un Archivo
 - a. Crea un archivo simple, por ejemplo, "mi-archivo.txt".

PROGRAMACIÓN I

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



```
MINGW64:/c/Users/Usuario/Documents/Emilita/TUP a Distancia/1° Semestr...
Usuario@DESKTOP-V0SKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2
$ git clone https://github.com/GomezJEmilia/Ejercicio-2.git
Cloning into 'Ejercicio-2'...
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.

Usuario@DESKTOP-V0SKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2
$
```

PROGRAMACIÓN I

```
MINGW64:/c/Users/Usuario/Documents/Emilita/TUP a Distancia/1° Semestr...
Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2
$ git add .
fatal: not a git repository (or any of the parent directories): .git

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2
$ cd Ejercicio-2/

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2/Ejercicio-2 (main)
$ git add .

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2/Ejercicio-2 (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)
    new file:   mi-archivo.txt

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2/Ejercicio-2 (main)
$ git commit -m "Agregando mi-archivo.txt"
[main 6b0b687] Agregando mi-archivo.txt
 1 file changed, 1 insertion(+)
 create mode 100644 mi-archivo.txt

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2/Ejercicio-2 (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 329 bytes | 329.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/GomezJEmilia/Ejercicio-2.git
 6538db7..6b0b687  main -> main

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2/Ejercicio-2 (main)
$ |
```

- Creando Branchs
 - a. Crear una Branch
 - b. Realizar cambios o agregar un archivo
 - c. Subir la Branch

PROGRAMACIÓN I

```
MINGW64:/c:/Users/Usuario/Documents/Emilita/TUP a Distancia/1° Semestr...
Programación I/Tareas y TPs/Trabajo Práctico n°2/Ejercicio-2 (main)
$ git checkout -b new-branch
Switched to a new branch 'new-branch'

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2/Ejercicio-2 (new-branch)
$ git commit -m "Agregando nuevo-archivo.txt"
On branch new-branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        archivo-nuevo.txt

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

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2/Ejercicio-2 (new-branch)
$ git add .

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2/Ejercicio-2 (new-branch)
$ git status
On branch new-branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   archivo-nuevo.txt

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2/Ejercicio-2 (new-branch)
$ git commit -m "Agregando nuevo-archivo.txt"
[new-branch 7788a04] Agregando nuevo-archivo.txt
1 file changed, 1 insertion(+)
 create mode 100644 archivo-nuevo.txt

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/
Programación I/Tareas y TPs/Trabajo Práctico n°2/Ejercicio-2 (new-branch)
$ git push origin new-branch
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 367 bytes | 183.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'new-branch' on GitHub by visiting:
remote:   https://github.com/GomezJEmilia/Ejercicio-2/pull/new/new-branch
remote:
To https://github.com/GomezJEmilia/Ejercicio-2.git
 * [new branch]      new-branch -> new-branch
```

PROGRAMACIÓN I

Ejercicio-2 Public

new-branch had recent pushes 2 minutes ago [Compare & pull request](#)

main 2 Branches 0 Tags [Add file](#) [Code](#)

GomezJEmilia Agregando mi-archivo.txt 6b0b687 · 8 minutes ago 2 Commits

README.md	Initial commit	16 minutes ago
mi-archivo.txt	Agregando mi-archivo.txt	8 minutes ago

README

Ejercicio-2

Repositorio para TP n° 2 - Ejercicio 2. Programación 1 - TUP

About

Repositorio para TP n° 2 - Ejercicio 2. Programación 1 - TUP

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published [Create a new release](#)

Packages

No packages published [Publish your first package](#)

Ejercicio-2 Public

new-branch had recent pushes 2 minutes ago [Compare & pull request](#)

new-branch 2 Branches 0 Tags [Add file](#) [Code](#)

This branch is 1 commit ahead of main. [Contribute](#)

GomezJEmilia Agregando nuevo-archivo.txt 7788a04 · 3 minutes ago 3 Commits

README.md	Initial commit	17 minutes ago
archivo-nuevo.txt	Agregando nuevo-archivo.txt	3 minutes ago
mi-archivo.txt	Agregando mi-archivo.txt	9 minutes ago

About

Repositorio para TP n° 2 - Ejercicio 2. Programación 1 - TUP

Readme

Activity

0 stars

1 watching

0 forks

Releases

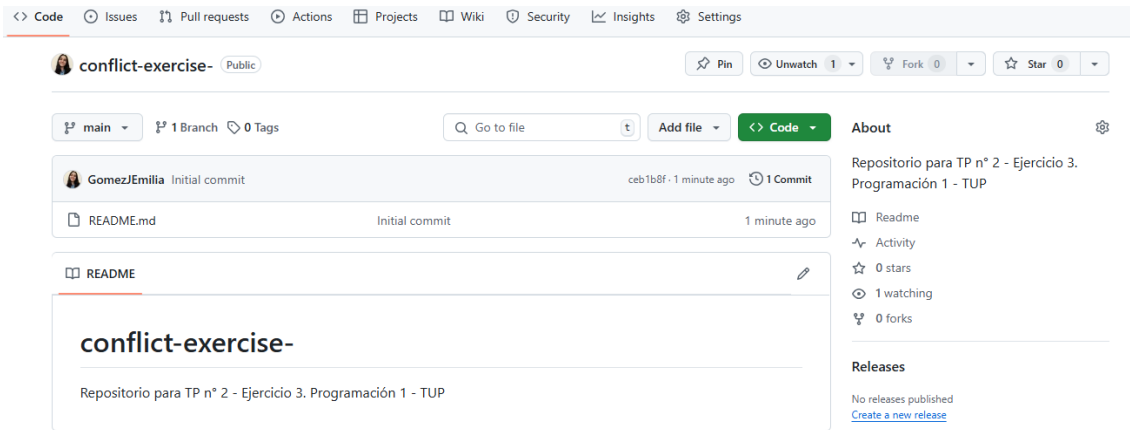
No releases published [Create a new release](#)

1) 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".

PROGRAMACIÓN I



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`

```
MINGW64~/Documents/Emilita/TUP a Distancia/1° Semestre/Programación I/Tareas y TPs/Trabajo Práctico n°2
$ git clone https://github.com/GomezJEmilia/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.

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/Programación I/Tareas y TPs/Trabajo Práctico n°2
$ cd conflict-exercise-/

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/Programación I/Tareas y TPs/Trabajo Práctico n°2/conflict-exercise- (main)
$
```

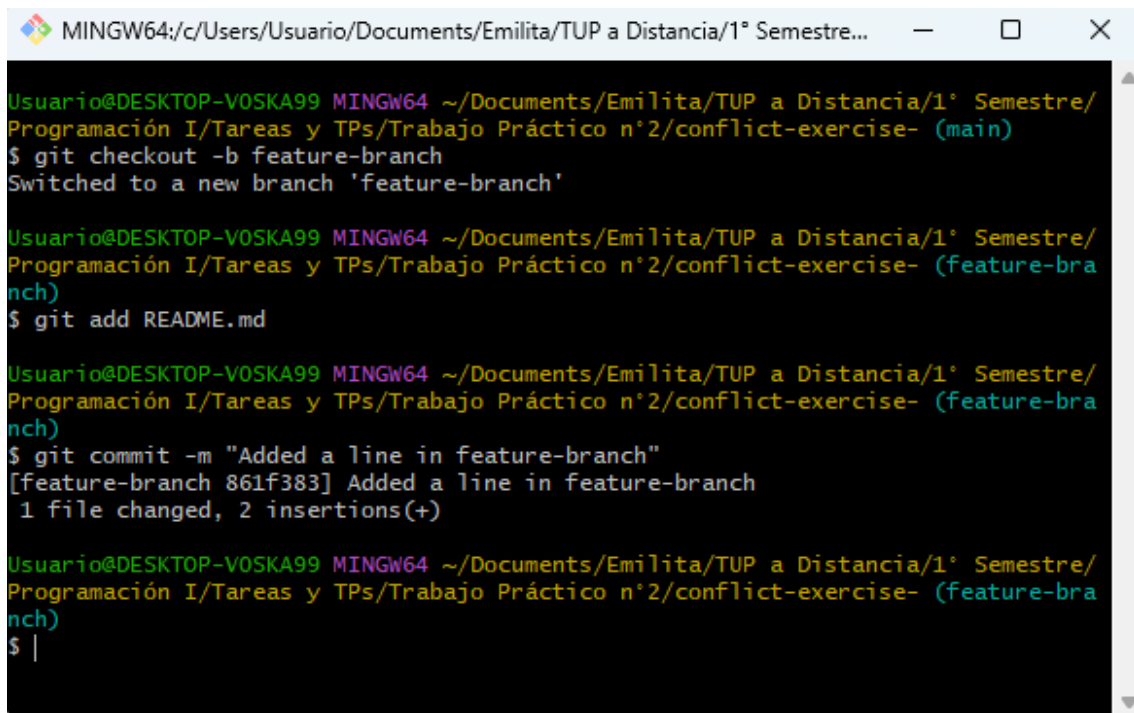
PROGRAMACIÓN I

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"`



```
MINGW64:/c:/Users/Usuario/Documents/Emilita/TUP a Distancia/1° Semestre/Programación I/Tareas y TPs/Trabajo Práctico n°2/conflict-exercise- (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/Programación I/Tareas y TPs/Trabajo Práctico n°2/conflict-exercise- (feature-branch)
$ git add README.md

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/Programación I/Tareas y TPs/Trabajo Práctico n°2/conflict-exercise- (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 861f383] Added a line in feature-branch
1 file changed, 2 insertions(+)

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/Programación I/Tareas y TPs/Trabajo Práctico n°2/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"`

PROGRAMACIÓN I

```
MINGW64/c:/Users/Usuario/Documents/Emilita/TUP a Distancia/1° Semestre/P...
Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/P
rogramación I/Tareas y TPs/Trabajo Práctico n°2/conflict-exercise- (feature-branc
h)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/P
rogramación I/Tareas y TPs/Trabajo Práctico n°2/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")

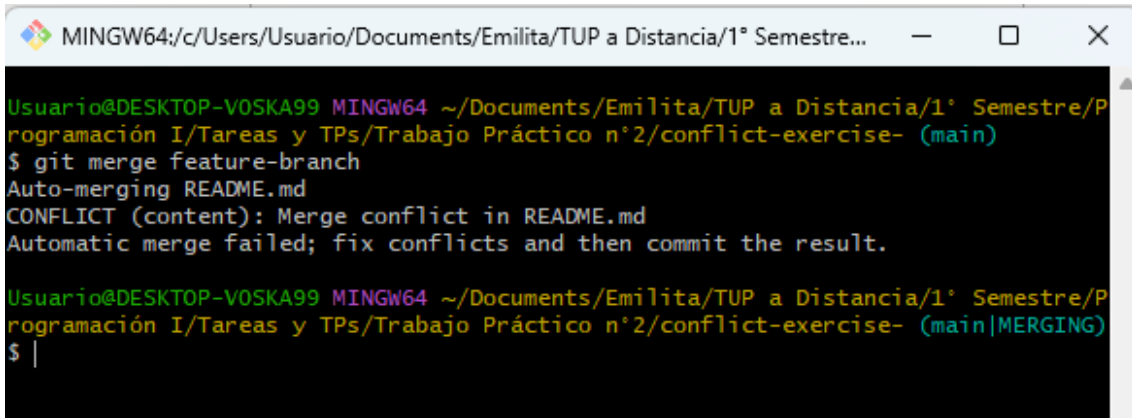
Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/P
rogramación I/Tareas y TPs/Trabajo Práctico n°2/conflict-exercise- (main)
$ git add README.md

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/P
rogramación I/Tareas y TPs/Trabajo Práctico n°2/conflict-exercise- (main)
$ git commit -m "Added a line in main branch"
[main aa5695c] Added a line in main branch
 1 file changed, 2 insertions(+)
```

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.

PROGRAMACIÓN I



```
MINGW64:/c:/Users/Usuario/Documents/Emilita/TUP a Distancia/1° Semestre...
Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/P
rogramación I/Tareas y TPs/Trabajo Práctico n°2/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.

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/P
rogramación I/Tareas y TPs/Trabajo Práctico n°2/conflict-exercise- (main|MERGING)
$ |
```

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"`

PROGRAMACIÓN I

```
1 # conflict-exercise-
2 Repositorio para TP nº 2 - Ejercicio 3. Programación 1 - TUP
3
4 Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
5 <<<<<<< HEAD (Current Change)
6 Este es un cambio en la main branch
7 =====
8 Este es un cambio en la feature branch.
9 >>>>>>> feature-branch (Incoming Change)
```

README.md X

C: > Users > Usuario > Documents > Emilita > TUP a Distancia > 1° Semestre > Programación I >

You, 19 seconds ago | 2 authors (You and one other)

```
1 # conflict-exercise-
2 Repositorio para TP nº 2 - Ejercicio 3. Programación 1 - TUP
3
4 Este es un cambio en la main branch y en la feature branch.
5
```

MINGW64:/c:/Users/Usuario/Documents/Emilita/TUP a Distancia/1° Semestre...

```
Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/P
rogramación I/Tareas y TPs/Trabajo Práctico nº2/conflict-exercise- (main|MERGING)
$ git add README.md

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/P
rogramación I/Tareas y TPs/Trabajo Práctico nº2/conflict-exercise- (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main 6763228] Resolved merge conflict

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/P
rogramación I/Tareas y TPs/Trabajo Práctico nº2/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`

PROGRAMACIÓN I

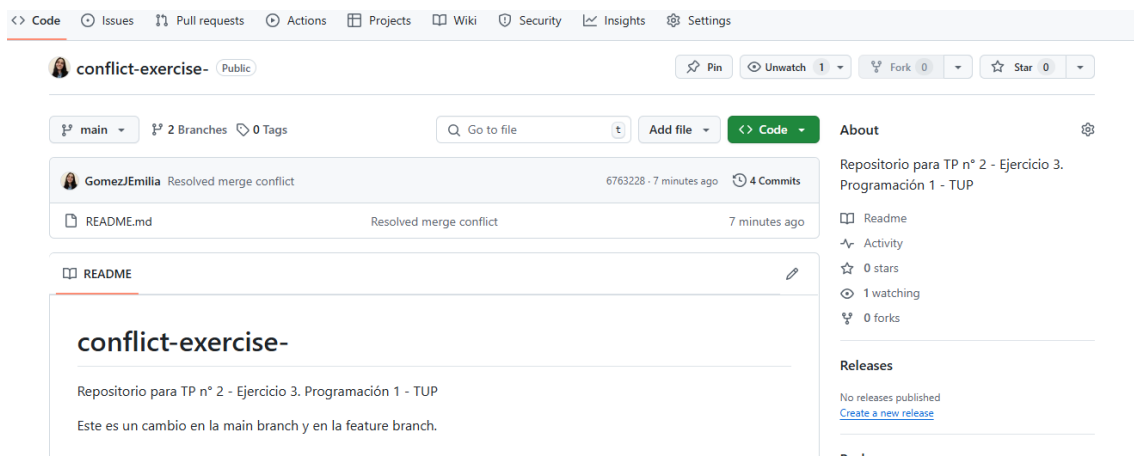
```
MINGW64;c:/Users/Usuario/Documents/Emilita/TUP a Distancia/1° Semestre/P
Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/P
rogramación I/Tareas y TPs/Trabajo Práctico n°2/conflict-exercise- (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 823 bytes | 205.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/GomezJEmilia/conflict-exercise-.git
   ceb1b8f..6763228  main -> main

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/P
rogramación I/Tareas y TPs/Trabajo Práctico n°2/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/GomezJEmilia/conflict-exercise-/pull/new/feature-
branch
remote:
To https://github.com/GomezJEmilia/conflict-exercise-.git
 * [new branch]      feature-branch -> feature-branch

Usuario@DESKTOP-VOSKA99 MINGW64 ~/Documents/Emilita/TUP a Distancia/1° Semestre/P
rogramación I/Tareas y TPs/Trabajo Práctico n°2/conflict-exercise- (main)
$
```












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.



PROGRAMACIÓN I

Commits

 main ▾	 All users ▾	 All time ▾
Commits on Mar 27, 2025		
<div>Resolved merge conflict</div> <div> GomezJEmilia committed 9 minutes ago</div> <div>6763228  <></div>		
<div>Added a line in main branch</div> <div> GomezJEmilia committed 1 hour ago</div> <div>aa5695c  <></div>		
<div>Added a line in feature-branch</div> <div> GomezJEmilia committed 1 hour ago</div> <div>861f383  <></div>		
<div>Initial commit</div> <div> GomezJEmilia authored 1 hour ago</div> <div>Verified ceb1b8f  <></div>		