

Práctico 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.

Actividades

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

- ¿Qué es GitHub?

GitHub es una plataforma basada en la nube donde se puede guardar, compartir y trabajar junto con otros usuarios para escribir código.

- ¿Cómo crear un repositorio en GitHub?

Primero hay que ir a la pagina de github (<https://github.com/>) y registrarse en ella. Luego en la esquina derecha hay un icono con un "+", se despliega y se elige la opción "New Repository". Se completa el nombre del repositorio. Se elige si el repositorio va a ser publico o privado. Y se clikea en el botón "Create repository".

- ¿Cómo crear una rama en Git?

Dentro de git se pone el comando "git branch" mas el nombre de la rama.

- ¿Cómo cambiar a una rama en Git?

Se usa el comando "git checkout" mas el nombre de la rama.

- ¿Cómo fusionar ramas en Git?

Nos ubicamos en la rama principal (a la que queremos sumarle la otra rama), por ejemplo la rama "master" o "main" con el siguiente comando "git checkout main" y luego "git merge (nombre de la otra rama que le quieron incorporar)".

- ¿Cómo crear un commit en Git?

Un commit se usa para guardar cambios (una instantánea del estado actual del proyecto). Dentro de git uso el comando "git commit -m (mensaje que quiero agregar)"

- ¿Cómo enviar un commit a GitHub?

Hacemos por ej una modificación del código. Luego en Git nos posicionamos en la rama que queremos estar (git Branch), nos fijamos que haya archivos modificados (git status), los agregamos (git add . o git add mas archivo especifico). Se usa el comando “git commit -m (mensaje que quiero agregar)” y luego hay que pusharlo hacia el repositorio de github, con el siguiente comando “git push add origin (nombre de la rama ej: master)”.

- ¿Qué es un repositorio remoto?

Es el proyecto que está guardado en la nube o en un servidor en línea por ej: en github.

- ¿Cómo agregar un repositorio remoto a Git?

Hay que abrir una terminal en git o powershell o una terminal en VSC y se pone el comando, vamos a la carpeta donde tenemos el proyecto local y luego usamos el comando “git add origin (nombre del repositorio)”-

- ¿Cómo empujar cambios a un repositorio remoto?

Nos ubicamos en la carpeta de nuestro proyecto. Se anota “git add .” y luego “git commit -m (mensaje de modificaciones)”. Por último, usamos el comando “git push add origin (nombre de la rama)”

- ¿Cómo tirar de cambios de un repositorio remoto?

Abrimos git en la carpeta de nuestro proyecto local. Luego “git pull origin (nombre de la rama)”.

Con pull traemos los cambios del repositorio remoto a nuestro repositorio local.

- ¿Qué es un fork de repositorio?

Es una copia de un repositorio ya existente. Esta copia en nuestro github por ejemplo nos permite modificar el repositorio sin hacer cambios en el repositorio original.

- ¿Cómo crear un fork de un repositorio?

Vamos al repositorio que nos interesa copiar, y arriba a la derecha hay un botón que dice “fork” y se hace una copia de este repositorio en nuestra cuenta de github.

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
Esto es para proponer cambios en un repositorio original. Vamos a nuestro fork en github y seleccionamos la opción “pull request”. Luego completamos la descripción detallando los cambios que hemos realizado. Una vez finalizado hacemos click en el botón “Create pull request”.

- ¿Cómo aceptar una solicitud de extracción?

El dueño del repositorio original revisa los cambios sugeridos y de aceptarlos los fusiona (botón “merge pull request”).

- ¿Qué es un etiqueta en Git?

También llamada *tag* es una referencia estática a un commit específico en el historial del repositorio. Se usan principalmente para marcar puntos importantes en el desarrollo de un proyecto.

- ¿Cómo crear una etiqueta en Git?

Se usa el comando “git tag (nombre de la etiqueta)”. O si se quiere para un commit anterior “git tag -a nombredelaetiqueta <hash_del_commit> -m "mensaje opcional”.

- ¿Cómo enviar una etiqueta a GitHub?

Usamos en git el comando “git push add origin (nombre de la etiqueta)”.

- ¿Qué es un historial de Git?

Es un registro detallado de todos los cambios realizados en un repositorio a lo largo del tiempo. Contiene información sobre cada commit realizado, incluyendo quién lo hizo, cuándo se hizo, y qué cambios específicos se implementaron en el código o archivos del repositorio.

- ¿Cómo ver el historial de Git?

Con el comando “git log” vemos una lista de todos los commits realizados en el repositorio, desde el más reciente hasta el más antiguo.

- ¿Cómo buscar en el historial de Git?

Se usa el comando “git log (mas filtro)”. Hay varios tipos de filtro, algunos de ellos:

git log --grep="mensaje a buscar" esto es para buscar un termino en los commits;

git log --author="nombre del autor" para buscar commits realizados por un autor específico;

git log --since="fecha_inicio" --until="fecha_fin" para buscar commits dentro de un rango de fecha;

git log -- <nombre_del_archivo> para buscar commits sobre un archivo específico;

git log <hash_del_commit> cuando se tiene el hash de un commit y se quiere buscar;

git log -p -- <nombre_del_archivo> para buscar y mostrar diferencias en los commits de un archivo;

git blame <nombre_del_archivo> para ver quien fue la ultima persona que modifico un archivo específico.

- ¿Cómo borrar el historial de Git?

Con el siguiente comando “git filter-branch --force --index-filter 'git rm --cached --ignore-unmatch <archivo_a_eliminar>' --prune-empty --tag-name-filter cat -- --all” se eliminan todos los commits y archivos del repositorio.

- ¿Qué es un repositorio privado en GitHub?

Es un repositorio cuyo acceso está restringido a un grupo selecto de personas, (solo los usuarios autorizados pueden ver, clonar o contribuir al código y otros archivos del repositorio).

- ¿Cómo crear un repositorio privado en GitHub?

Dentro de github seleccionar la opción crear un nuevo repositorio, se le da un nombre y antes de aceptar se selecciona la opción “privado”.

- ¿Cómo invitar a alguien a un repositorio privado en GitHub?

En la pagina principal de nuestro repositorio hay que ir a la pestaña “Settings”. En la barra lateral izquierda elegir “Manage Access”. Luego “Invite a collaborator” Aparecerá un cuadro de texto donde se puede buscar el nombre de usuario de la persona a la que se desee invitar.

- ¿Qué es un repositorio público en GitHub?

Es un repositorio cuyo contenido es visible para todos sin necesidad de permisos especiales.

- ¿Cómo crear un repositorio público en GitHub?

Al momento de crear un repositorio, luego de elegir el nombre del mismo y antes de aceptar la creación del mismo se selecciona la opción “Publico”.

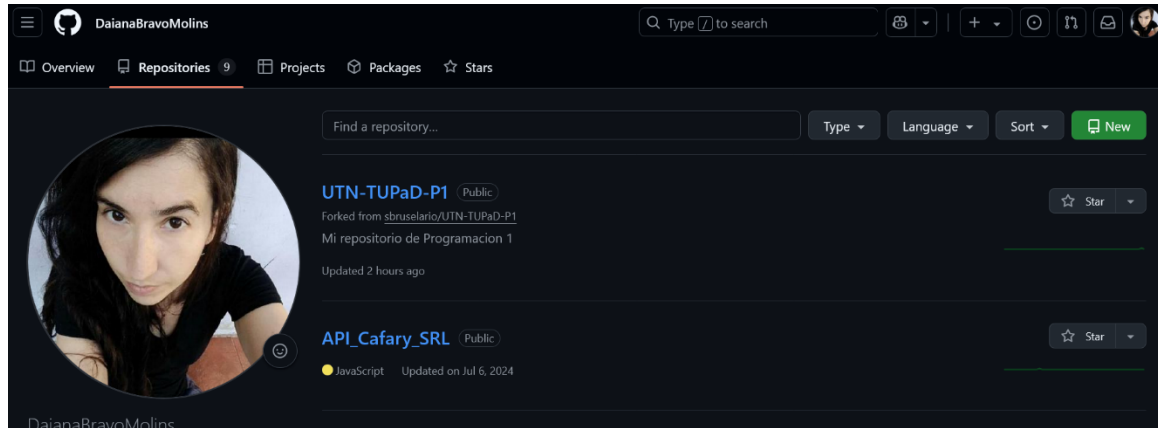
- ¿Cómo compartir un repositorio público en GitHub?

Ubicados en la pagina principal de nuestro repositorio copiamos la “URL”.

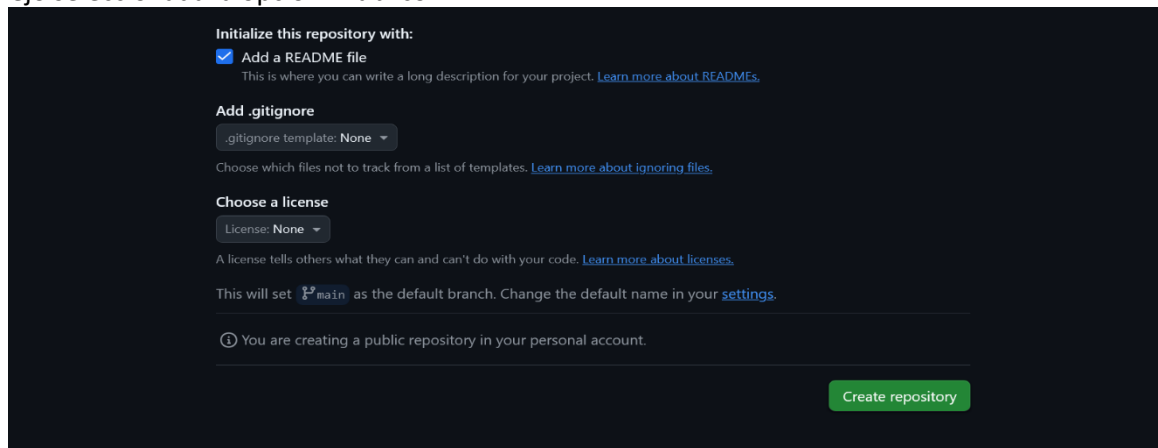
2) Realizar la siguiente actividad:

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

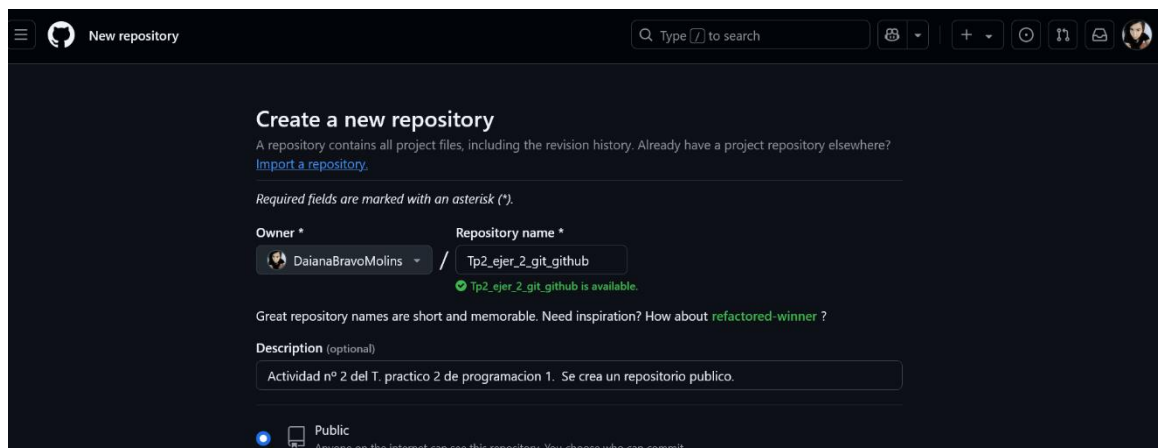
Ingreso a mi cuenta en github:



Dejo seleccionada la opción “Publico”:

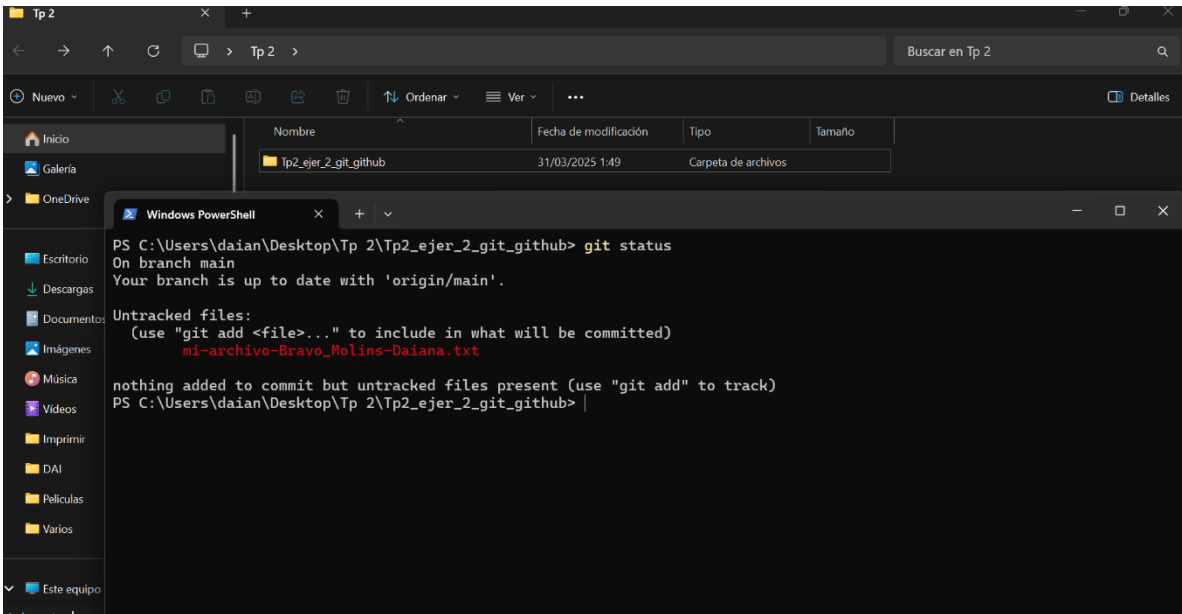


Elijo un nombre para el repositorio:



- 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).

mi-archivo:



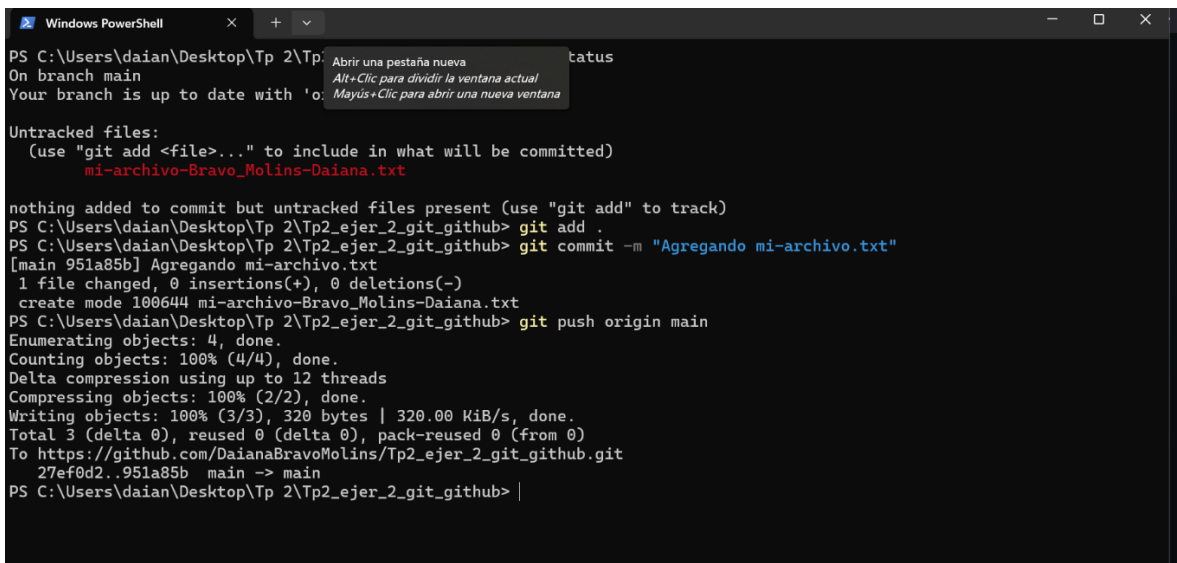
The screenshot shows a Windows File Explorer window with the address bar set to 'Tp2'. The left sidebar shows the 'OneDrive' section. The main pane displays a folder named 'Tp2_ejer_2_git_github' with a modification date of '31/03/2025 1:49' and a type of 'Carpeta de archivos'. Overlaid on the File Explorer is a Windows PowerShell terminal window. The terminal shows the following commands and output:

```
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        mi-archivo-Bravo_Molins-Daiana.txt

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> |
```

Cargo el archivo a github:



The screenshot shows a Windows PowerShell terminal window with the following commands and output:

```
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        mi-archivo-Bravo_Molins-Daiana.txt

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git add .
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git commit -m "Agregando mi-archivo.txt"
[main 951a85b] Agregando mi-archivo.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 mi-archivo-Bravo_Molins-Daiana.txt
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 320 bytes | 320.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/DaianaBravoMolins/Tp2_ejer_2_git_github.git
 27ef0d2..951a85b  main -> main
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> |
```

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

Se crea la rama secundaria:

```
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
      mi-archivo-Bravo_Molins-Daiana.txt

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git add .
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git commit -m "Agregando mi-archivo.txt"
[main 951a85b] Agregando mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo-Bravo_Molins-Daiana.txt
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 320 bytes | 320.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/DaianaBravoMolins/Tp2_ejer_2_git_github.git
   27ef0d2..951a85b  main -> main
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git branch rama_secundaria
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> |
```

Subo el archivo txt a github:

```
Windows PowerShell
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git add "archivo_secundario.txt"
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git commit -m "Se carga rama secundaria en archivo txt"
[main 12334a9] Se carga rama secundaria en archivo txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 archivo_secundario.txt
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 337 bytes | 337.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/DaianaBravoMolins/Tp2_ejer_2_git_github.git
   951a85b..12334a9  main -> main
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> |
```


Como cargue el txt en la rama principal en lugar de la secundaria, corrijo el error:

```
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git branch
* main
  rama_secundaria
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git checkout rama_secundaria
Switched to branch 'rama_secundaria'
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git checkout main -- archivo_secundario.txt
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git add archivo_secundario.txt
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git commit -m "muevo el archivo_secundario a la rama alternativa que cree anteriormente"
[rama_secundaria a28a9f6] muevo el archivo_secundario a la rama alternativa que cree anteriormente
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 archivo_secundario.txt
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git push origin rama_secundaria
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 358 bytes | 358.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'rama_secundaria' on GitHub by visiting:
remote:   https://github.com/DaianaBravoMolins/Tp2_ejer_2_git_github/pull/new/rama_secundaria
remote:
To https://github.com/DaianaBravoMolins/Tp2_ejer_2_git_github.git
 * [new branch]      rama_secundaria -> rama_secundaria
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git rm archivo_secundario.txt
rm 'archivo_secundario.txt'
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git commit -m "Se elimino archivo secundario de la rama principal puesto que se debia haber cargado en la rama alternativa"
[main 77212bf] Se elimino archivo secundario de la rama principal puesto que se debia haber cargado en la rama alternativa
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git commit -m "Se elimino archivo secundario de la rama principal puesto que se debia haber cargado en la rama alternativa"
[main 77212bf] Se elimino archivo secundario de la rama principal puesto que se debia haber cargado en la rama alternativa
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 archivo_secundario.txt
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 292 bytes | 292.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/DaianaBravoMolins/Tp2_ejer_2_git_github.git
 12334a9..77212bf  main -> main
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> |
```

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".

Owner * DaianaBravoMolins / Repository name * conflict-exercise
✔ conflict-exercise is available.

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

Description (optional)
Ejercicio 3 del Segundo TP de Programacion 1

☒ 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](#).

i 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
```

```
Windows PowerShell
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> cd ..
PS C:\Users\daian\Desktop\Tp 2> cd ..
PS C:\Users\daian\Desktop> git clone https://github.com/DaianaBravoMolins/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.
PS C:\Users\daian\Desktop> cd conflict
PS C:\Users\daian\Desktop> cd conflict-exercise
PS C:\Users\daian\Desktop\conflict-exercise> |
```

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"
```

```
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> cd ..
PS C:\Users\daian\Desktop\Tp 2> cd ..
PS C:\Users\daian\Desktop> git clone https://github.com/DaianaBravoMolins/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.
PS C:\Users\daian\Desktop> cd conflict
PS C:\Users\daian\Desktop> cd conflict-exercise
PS C:\Users\daian\Desktop\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\daian\Desktop\conflict-exercise> git add README.md
PS C:\Users\daian\Desktop\conflict-exercise> git commit -m "Se agrego una linea en la rama feature"
[feature-branch a70de87] Se agrego una linea en la rama feature
1 file changed, 2 insertions(+)
PS C:\Users\daian\Desktop\conflict-exercise> |
```

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"`

```
PS C:\Users\daian\Desktop\Tp 2\Tp2_ejer_2_git_github> cd ..
PS C:\Users\daian\Desktop\Tp 2> cd ..
PS C:\Users\daian\Desktop>
PS C:\Users\daian\Desktop> git clone https://github.com/DaianaBravoMolins/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.
PS C:\Users\daian\Desktop> cd conflict\C
PS C:\Users\daian\Desktop> cd conflict-exercise
PS C:\Users\daian\Desktop\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\daian\Desktop\conflict-exercise> git add README.md
PS C:\Users\daian\Desktop\conflict-exercise> git commit -m "Se agrego una linea en la rama feature"
[feature-branch a70de87] Se agrego una linea en la rama feature
 1 file changed, 2 insertions(+)
PS C:\Users\daian\Desktop\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\daian\Desktop\conflict-exercise> git add README.md
PS C:\Users\daian\Desktop\conflict-exercise> git commit -m "Se agrego una linea diferente en la rama main"
[main dd0146b] Se agrego una linea diferente en la rama main
 1 file changed, 3 insertions(+)
PS C:\Users\daian\Desktop\conflict-exercise> |
```

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.

```
PS C:\Users\daian\Desktop\conflict-exercise> git branch
  feature-branch
* main
PS C:\Users\daian\Desktop\conflict-exercise> 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.
PS C:\Users\daian\Desktop\conflict-exercise> |
```

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"
```

```
PS C:\Users\daian\Desktop\conflict-exercise> git branch
feature-branch
* main
PS C:\Users\daian\Desktop\conflict-exercise> 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.
PS C:\Users\daian\Desktop\conflict-exercise> git add README.md
PS C:\Users\daian\Desktop\conflict-exercise> git commit -m "Arreglo el conflicto que se produjo al fusionar ramas."
gid : El término 'gid' no se reconoce como nombre de un cmdlet, función, archivo de script o programa ejecutable.
Compruebe si escribió correctamente el nombre o, si incluyó una ruta de acceso, compruebe que dicha ruta es correcta e
inténtelo de nuevo.
En línea: 1 Carácter: 1
+ git commit -m "Arreglo el conflicto que se produjo al fusionar ramas. ...
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (gid:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\daian\Desktop\conflict-exercise> git commit -m "Arreglo el conflicto que se produjo al fusinar ramas."
[main c9602d4] Arreglo el conflicto que se produjo al fusinar ramas.
PS C:\Users\daian\Desktop\conflict-exercise> |
```

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

```
PS C:\Users\daian\Desktop\conflict-exercise> git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 1000 bytes | 250.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/DaianaBravoMolins/conflict-exercise.git
  3c0ecac..c9602d4  main -> main
PS C:\Users\daian\Desktop\conflict-exercise> git push origin feature
error: src refspec feature does not match any
error: failed to push some refs to 'https://github.com/DaianaBravoMolins/conflict-exercise.git'
PS C:\Users\daian\Desktop\conflict-exercise> git push feature-branch
fatal: 'feature-branch' does not appear to be a git repository
fatal: Could not read from remote repository.

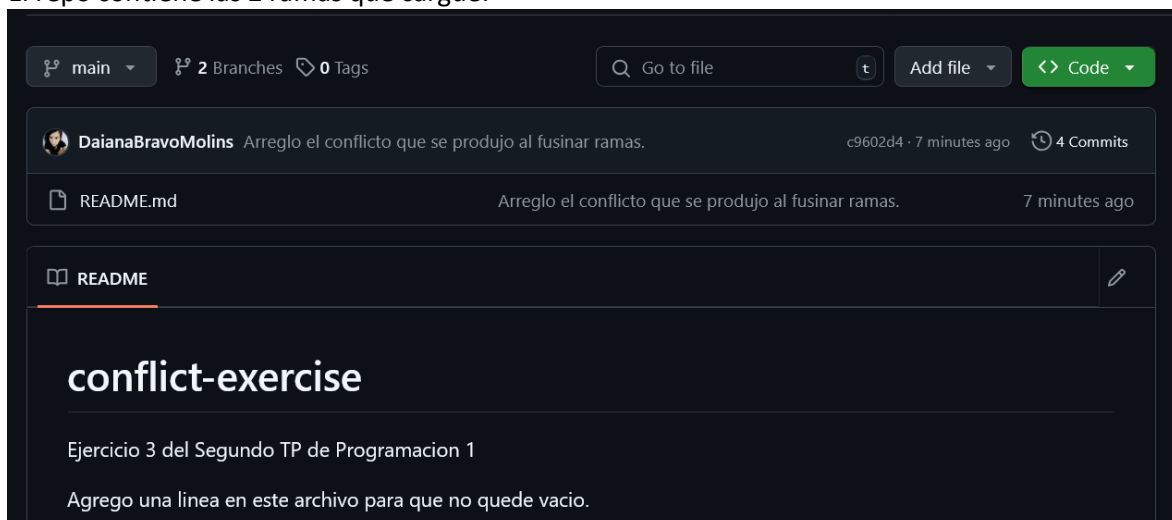
Please make sure you have the correct access rights
and the repository exists.
PS C:\Users\daian\Desktop\conflict-exercise> git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/DaianaBravoMolins/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/DaianaBravoMolins/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch
PS C:\Users\daian\Desktop\conflict-exercise> |
```

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.

Verificación:

El repo contiene las 2 ramas que cargue:



El README quedo modificado:



The screenshot shows a GitHub interface for a repository named 'conflict-exercise'. The file 'README.md' is selected in the left sidebar. The main area displays a commit by user 'DaianaBravoMolins' with the message 'Arreglo el conflicto que se produjo al fusinar ramas.' The commit hash is 'c9602d4' and it was made '6 minutes ago'. The commit details show '7 lines (3 loc) · 127 Bytes' and a note 'Code 55% faster with GitHub Copilot'. The 'Code' tab is active, showing the following content:

```
1 # conflict-exercise
2 Ejercicio 3 del Segundo TP de Programacion 1
3
4
5
6 Agrego una linea en este archivo para que no quede vacio.
```