

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?

comunidad donde podemos compartir nuestros repositorios publica o privada

¿Cómo crear un repositorio en GitHub?

Git innit

¿Cómo crear una rama en Git?

Git branch

¿Cómo cambiar a una rama en Git?

Git checkout

¿Cómo fusionar ramas en Git?

Git merge

¿Cómo crear un commit en Git?

Git commit – m “ mensaje”

¿Cómo enviar un commit a GitHub?

git push origin nombre_de_la_rama

¿Qué es un repositorio remoto?

Sitio donde se almacena nuestros proyectos nosotros usamos git hub

¿Cómo agregar un repositorio remoto a Git?

Desde nuestra terminal local hacemos

Git remote add origin url_repositorio

¿Cómo empujar cambios a un repositorio remoto?

obtener los últimos cambios del repositorio remoto para evitar conflictos:

git pull origin nombre_de_la_rama

empuja tus cambios al repositorio remoto:

git push origin nombre_de_la_rama

¿Cómo tirar de cambios de un repositorio remoto?

git pull origin nombre_de_la_rama

lo utilizamos para descargar y fusionar los cambios del repositorio remoto con la rama local.

¿Qué es un fork de repositorio?

Son proyectos compartidos en el sitio de git hub para uso propio o para la comunidad, podemos disponer de una copia en nuestro repositorio si queremos usar el proyecto ed otra persona

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

Para hacer el pull request nos dirigiremos a la solapa de Pull requests allí daremos click en new pull request, veremos una ventana a modo de resumen en donde se reflejarán los cambios que hemos hecho nosotros en comparación al repositorio original (el código original, mejor dicho). Daremos click en Create pull request donde veremos el asunto (colocamos algún mensaje global) y más abajo tenemos suficiente lugar para poder explayarnos en mencionar el porque ese cambio que hemos realizado nosotros, sería considerado como algo que a el repositorio original le vendrían bien agregarlo.

¿Cómo aceptar una solicitud de extracción?

El autor del repositorio verá en sus pull requests el mensaje que le hemos enviado, para que lo pueda observar y si lo considera realizar el cambio pertinente (además de poder responderle al usuario que le ha propuesto ese cambio). Lo bueno de todo esto es que si el usuario original considera que esta modificación

Es buena y no genera conflictos con la rama maestra de su repositorio local remoto, puede clickear en Merge pull request y de esta manera sumará a su repositorio los cambios que hizo un usuario (en modo de ayuda).

¿Qué es un etiqueta en Git?

Como muchos VCS, Git tiene la posibilidad de etiquetar puntos específicos del historial como importantes.

¿Cómo crear una etiqueta en Git?

Git utiliza dos tipos principales de etiquetas: ligeras y anotadas.

Una etiqueta ligera es muy parecida a una rama que no cambia - simplemente es un puntero a un commit específico.

Sin embargo, las etiquetas anotadas se guardan en la base de datos de Git como objetos enteros. Tienen un checksum; contienen el nombre del etiquetador, correo electrónico y fecha; tienen un mensaje asociado; y pueden ser firmadas y verificadas con GNU Privacy Guard (GPG). Normalmente se recomienda que crees etiquetas anotadas, de manera que

tengas toda esta información; pero si quieres una etiqueta temporal o por alguna razón no estás interesado en esa información, entonces puedes usar las etiquetas ligeras.

¿Cómo enviar una etiqueta a GitHub?

Primero, debes crear una etiqueta en tu repositorio local. Puedes crear una etiqueta anotada (que incluye información adicional como el autor y la fecha) o una etiqueta ligera (simplemente un puntero a un commit):

Ej. etiqueta ligera: `git tag v1.0`

Podes verificar las etiquetas que has creado localmente con:

```
git tag
```

Una vez que has creado la etiqueta en tu repositorio local, necesitas empujarla al repositorio remoto en GitHub. Puedes hacer esto con el siguiente comando: `git push origin v1.0`

(origin es el nombre del repositorio remoto (por defecto suele ser origin) y v1.0 es el nombre de la etiqueta.) Para empujar todas las etiquetas creadas, usar:

```
git push origin --tags
```

¿Qué es un historial de Git?

El historial de Git es una secuencia de todos los cambios realizados en un repositorio de Git. Cada cambio en el repositorio se guarda como un commit, y cada commit contiene información sobre el estado del proyecto en un momento específico, incluyendo:

Identificador del commit

Autor

Fecha de realización

Mensaje enviado

¿Cómo ver el historial de Git?

Esto lo conseguimos con el comando de Git:

```
git log
```

Con tipear este comando en el bash de Git podremos apreciar el histórico de commits, estando situados en la carpeta de nuestro proyecto. El listado de commits estará invertido, es decir, los últimos realizados aparecen los primeros.

El comando `git log --oneline` es una forma compacta de visualizar el historial de commits en un repositorio Git. Muestra un resumen conciso de los commits recientes, con cada commit representado en una sola línea.

Si tu proyecto ya tiene muchos commits, quizás no quieras mostrarlos todos, ya que generalmente no querrás ver cosas tan antiguas como el origen del repositorio. Para ver un número de logs determinado introducimos ese número como opción, con el signo "-" delante (-1, -8, -12...). Por ejemplo, esto muestra los últimos tres commits: `git log -3`

Si queremos que el log también nos muestre los cambios en el código de cada commit podemos usar la opción `-p`. Esta opción genera una salida mucho más larga, por lo que seguramente nos tocará movernos en la salida con los cursores y usaremos `CTRL + Z` para salir. `git log -2 -p`

¿Cómo buscar en el historial de Git?

Para buscar en el historial de commits de Git, puedes utilizar varios comandos y opciones que te permiten filtrar y localizar commits específicos.

Para buscar commits que contengan una palabra o frase específica en el mensaje de commit, usa `git log` con la opción `-grep`: `git log --grep="palabra clave"`

Para buscar commits que han modificado un archivo específico, usa `git log` seguido del nombre del archivo: `git log -- nombre_del_archivo`

Para buscar commits en un rango de fechas específico, usa las opciones `--`

`since` y `--until`: `git log --since="2024-01-01" --until="2024-01-31"`

Para encontrar commits hechos por un autor específico, usa `--author`:

```
git log --author="Nombre del Autor"
```

¿Cómo borrar el historial de Git?

El comando git reset se utiliza sobre todo para deshacer las cosas, como posiblemente puedes deducir por el verbo. Se mueve alrededor del puntero HEAD y opcionalmente cambia el index o área de ensayo y también puede cambiar opcionalmente el directorio de trabajo si se utiliza - hard. Esta última opción hace posible que este comando pueda perder tu trabajo si se usa incorrectamente, por lo que asegúrese de entenderlo antes de usarlo.

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.

¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un tipo de repositorio en el que el contenido solo es accesible para usuarios específicos que han sido autorizados. A diferencia de los repositorios públicos, donde cualquier persona puede ver y clonar el contenido, un repositorio privado limita el acceso a los colaboradores que tú elijas. Esto es útil para proyectos que contienen información sensible o que aún están en desarrollo y no deseas que estén disponibles públicamente

¿Cómo crear un repositorio privado en GitHub?

Inicia sesión en GitHub

Ingresa a la página de creación de repositorios:

En la esquina superior derecha de la página principal, debes hacer clic en el botón “+” y seleccionar “New Repository” o hacer clic en “New”:

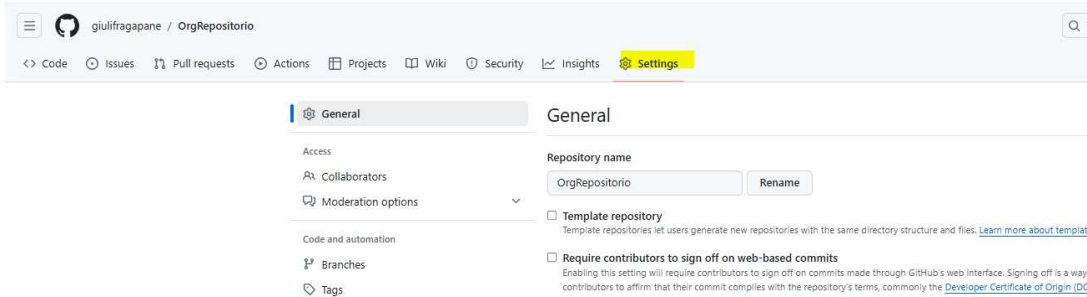
Completar la información del repositorio (nombre del repositorio, descripción) Seleccionar la configuración de privacidad:

Esto asegura que el repositorio será privado y solo accesible para los colaboradores que tú elijas.

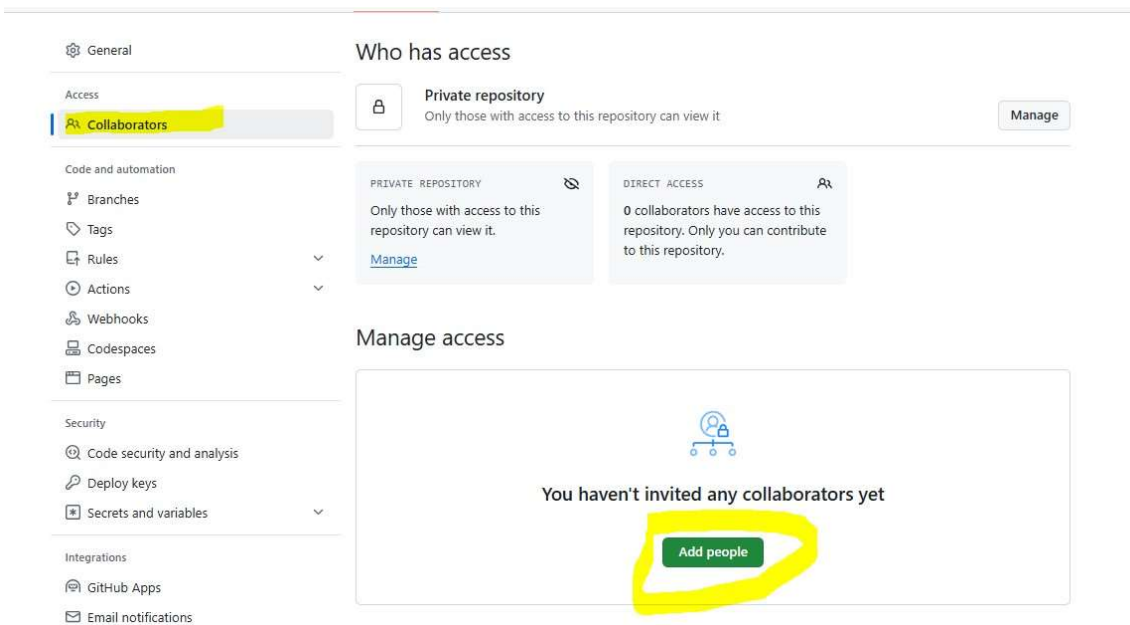
¿Cómo invitar a alguien a un repositorio privado en GitHub?

Invitar a alguien a un repositorio privado en GitHub es un proceso sencillo, pero requiere permisos adecuados.

Accede al repositorio, **haz clic en la pestaña "Settings"** del repositorio. Está en la parte superior del repositorio, junto a las pestañas como "Code" y "Issues"



Selecciona "Collaborators" en el menú de la izquierda. Esto te llevará a la página donde puedes administrar colaboradores.



En la sección "Collaborators", haz clic en el botón **"Add people"** e ingresa el nombre de usuario de GitHub de la persona que deseas invitar. Selecciona el nivel de acceso que deseas otorgar: **Read, Triage, Write, Maintain, o Admin**. Haz clic en el botón **"Add"** para enviar la invitación.

¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio cuyo contenido es accesible a cualquier persona en Internet. A diferencia de un repositorio privado, que está restringido a un grupo

específico de colaboradores, un repositorio público permite que cualquier persona pueda ver, clonar y, si tienen los permisos adecuados, contribuir al proyecto.

¿Cómo crear un repositorio público en GitHub?

Pasos:

Inicia sesión en GitHub

Ingresa a la página de creación de repositorios:

En la esquina superior derecha de la página principal, debes hacer clic en el botón “+” y seleccionar “New Repository” o hacer clic en “New”:

Completar la información del repositorio (nombre del repositorio, descripción)

Seleccionar la configuración de privacidad:

público

¿Cómo compartir un repositorio público en GitHub?

La forma más sencilla de compartir tu repositorio es proporcionar el enlace directo al mismo.

Accede a tu repositorio, copia la URL de tu repositorio que se encuentra en un cuadro de texto que dice "<> Code”:

2 Realizar la siguiente actividad:

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

Crear un nuevo repositorio

Un repositorio contiene todos los archivos del proyecto, incluido el historial de revisiones. ¿Ya tienes un repositorio de proyectos en otro lugar? [Importa un repositorio.](#)

Los campos obligatorios están marcados con un asterisco (*).

Dueño *



Wally-ux

Nombre del repositorio *

tp2_actividad2

✔ tp2_actividad está disponible.

Los buenos nombres de repositorios son cortos y fáciles de recordar. ¿Necesitas inspiración? ¿Qué te parece?
probable-palito de pescado ?

Descripción (opcional)

tp2_actividad2

☒ Público

Cualquier persona en internet puede ver este repositorio. Tú decides quién puede contribuir.

☐ Privado

Tú eliges quién puede ver y comprometerse con este 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).

Ejercicio sigue abajo, screen de pantalla

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.5608]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\agust>git clone https://github.com/Wally-ux/tp2_actividad2.git
Cloning into 'tp2_actividad2'...
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.

C:\Users\agust>cd tp2_actividad2

C:\Users\agust\tp2_actividad2>echo "mi_archivo wally"> mi-archivo.txt

C:\Users\agust\tp2_actividad2>git add .




C:\Users\agust\tp2_actividad2>git commit -m "Agregando mi-archivo.txt"
[main d2738e1] Agregando mi-archivo.txt
 1 file changed, 1 insertion(+)
 create mode 100644 mi-archivo.txt

C:\Users\agust\tp2_actividad2>git status
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

C:\Users\agust\tp2_actividad2>
```

Este equipo > Disco local (C:) > Usuarios > agust > tp2_actividad2 >

Nombre	Fecha de modificación	Tip
 .git	30/3/2025 23:45	Car
 mi-archivo	30/3/2025 23:43	Doc
 README	30/3/2025 23:33	Arc

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

```
C:\Users\agust\tp2_actividad2>git branch
* main

C:\Users\agust\tp2_actividad2>git push origin main
info: please complete authentication in your browser...
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), 315 bytes | 157.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Wally-ux/tp2_actividad2.git
 5636cd6..d2738e1  main -> main

C:\Users\agust\tp2_actividad2>git push origin main
Everything up-to-date
```

tp2_actividad2 Público

Alfiler Dejar de ver 1

principal 1 sucursal 0 etiquetas Ir al archivo t Agregar archivo <> Código

Wally-ux Agregando mi-archivo.txt d2738e1 · Hace 10 minutos 2 confirmaciones

README.md	Compromiso inicial	Hace 32 minutos
mi-archivo.txt	Agregando mi-archivo.txt	Hace 10 minutos

LÉAME

tp2_actividad2

tp2_actividad2

```
C:\Users\agust\tp2_actividad2>git push origin main
Everything up-to-date

C:\Users\agust\tp2_actividad2>git branch ramma

C:\Users\agust\tp2_actividad2>git branch
* main
  ramma

C:\Users\agust\tp2_actividad2>git checkout ramma
Switched to branch 'ramma'

C:\Users\agust\tp2_actividad2>git branch
  main
* ramma

C:\Users\agust\tp2_actividad2>_
```

```
ramma

C:\Users\agust\tp2_actividad2>echo "modificacion desde rama" > mi_archivo.txt

C:\Users\agust\tp2_actividad2>git add .

C:\Users\agust\tp2_actividad2>git status
On branch ramma
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   mi_archivo.txt

C:\Users\agust\tp2_actividad2>echo "modificacion desde rama" > mi-archivo.txt

C:\Users\agust\tp2_actividad2>git status
On branch ramma
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   mi_archivo.txt
```

```
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:   mi-archivo.txt

C:\Users\agust\tp2_actividad2>git commit -m "nueva modificacion rama"
[ramma dd593e8] nueva modificacion rama
 1 file changed, 1 insertion(+)
 create mode 100644 mi_archivo.txt

C:\Users\agust\tp2_actividad2>_
```


Archivo modificado desde rama


Subimos la nueva ramma



```
create mode 100644 mi_archivo.txt


:\Users\agust\tp2_actividad2>git push origin ramma
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), 344 bytes | 344.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'ramma' on GitHub by visiting:
remote:   https://github.com/Wally-ux/tp2_actividad2/pull/new/ramma
remote:
to https://github.com/Wally-ux/tp2_actividad2.git
* [new branch]      ramma -> ramma

:\Users\agust\tp2_actividad2>_
```


 **tp2_actividad2** Public

 **ramma** had recent pushes 1 minute ago


 main ▾

 2 Branches

 0 Tags

 Go to file

 **Wally-ux** Agregando mi-archivo.txt

 README.md Initial commit

 mi-archivo.txt Agregando mi-archivo.txt


Tengo en git 2 branches

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".

 **conflict-exercise.-** Public

 main  1 Branch  0 Tags

 Go to file

 Wally-ux Initial commit

 README.md

Initial commit

 README

conflict-exercise.-

ejercicio de conflicto

-
-
-

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>



```
C:\Users\agust\tp2_actividad2>git clone https://github.com/Wally-ux/conflict-exercise.-
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.
C:\Users\agust\tp2_actividad2>
```

- Entra en el directorio del repositorio: **cd conflict-exercise**

```
C:\Users\agust\tp2_actividad2>cd conflict-exercise
El sistema no puede encontrar la ruta especificada.

C:\Users\agust\tp2_actividad2>cd conflict-exercise.
El sistema no puede encontrar la ruta especificada.

C:\Users\agust\tp2_actividad2>cd conflict-exercise.-
C:\Users\agust\tp2_actividad2\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.

```
File Edit Selection Find View Goto Tools Project Preferences
< > README.md
1 #conflict-exercise
2 ponemos en prueba los conflictos
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"`

```
C:\Users\agust\tp2_actividad2\conflict-exercise.->git add README.md
C:\Users\agust\tp2_actividad2\conflict-exercise.->git commit -m "Added a line in feature-branch"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
C:\Users\agust\tp2_actividad2\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`


```
C:\Users\agust\tp2_actividad2\conflict-exercise.->git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.

C:\Users\agust\tp2_actividad2\conflict-exercise.->git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\agust\tp2_actividad2\conflict-exercise.->git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.

C:\Users\agust\tp2_actividad2\conflict-exercise.->git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.

C:\Users\agust\tp2_actividad2\conflict-exercise.->
```

No puedo cambiar a branch main..

```
C:\Users\agust\tp2_actividad2\conflict-exercise.->cd..

C:\Users\agust\tp2_actividad2>git status
On branch ramma
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:   mi-archivo.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        conflict-exercise.-/

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

C:\Users\agust\tp2_actividad2>git checkout main
M       mi-archivo.txt
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\agust\tp2_actividad2>git branch
* main
  ramma

C:\Users\agust\tp2_actividad2>
```

Dudas si esta bien?

- 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"`

```
C:\Users\agust\tp2_actividad2>git add README.md
C:\Users\agust\tp2_actividad2>git commit -m "Added a line in main branch"
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:   mi-archivo.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        conflict-exercise.-/

no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\agust\tp2_actividad2>_
```

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.

```
no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\agust\tp2_actividad2>git merge feature-branch
merge: feature-branch - not something we can merge
C:\Users\agust\tp2_actividad2>_
```

Al parecer no tengo conflicto?

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"
```

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
```

```
merge: feature-branch < not something we can merge
C:\Users\agust\tp2_actividad2>git push origin main
Everything up-to-date
C:\Users\agust\tp2_actividad2>git push origin feature-branch
error: src refspec feature-branch does not match any
error: failed to push some refs to 'https://github.com/Wally-ux/tp2_actividad2.git'
C:\Users\agust\tp2_actividad2>_
```


Error al hacer 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.

> Código Asuntos Solicitudes de extracción Comportamiento Proyectos

 ejercicio de conflicto.- Público

principal 1 sucursal 0 etiquetas Ir al archivo

 Wally-ux Compromiso inicial 6bl

README.md Compromiso inicial

LÉAME

ejercicio de conflicto.-

ejercicio de conflicto

