

Programación I

Trabajo Practico N°2

Actividades:

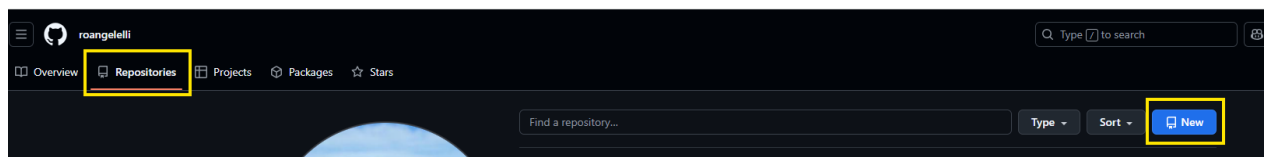
- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

a. **¿Qué es GitHub?**

Es una comunidad donde podremos compartir repositorios de forma publica y privada. Es una plataforma en línea, donde podremos almacenar, gestionar y controlar las versiones de diferentes códigos. Esta basada y conectada con Git para enlazar las versiones remotas y locales de nuestro código.

b. **¿Cómo crear un repositorio en GitHub?**

Para crear un repositorio en GitHub debemos dirigirnos a la sección “Repositories” y tocar en el botón “New”, luego de esto llenar algunos datos básicos que nos solicita la web antes de crearlo. (Nombre, descripción, privacidad, etc)



c. **¿Cómo crear una rama en Git?**

Para crear una rama en Git debemos hacerlo desde la consola y utilizar el comando:

git branch <nombreDeLaRama>

d. **¿Cómo cambiar a una rama en Git?**

Para cambiar a otra rama en Git podemos usar primeramente el comando ***git branch*** para ver los nombres de todas las ramas y luego utilizar el comando ***git checkout <nombreDeLaRama>*** a la que queremos cambiar.

e. **¿Cómo fusionar ramas en Git?**

Para fusionar ramas en Git, debemos pararnos en la rama main (o la que queramos fusionar) y ejecutar el comando ***git merge <nombreRamaAFusionarConActual>***

f. **¿Cómo crear un commit en Git?**

Para crear un commit en Git se utiliza el comando ***git commit***, pero si queremos hacerlo con un mensaje (recomendable) debemos utilizar ***git commit -m "mensaje"***

g. ¿Cómo enviar un commit a GitHub?

Para enviar un commit a GitHub debemos asegurarnos de primero haber realizado una serie de pasos:

1. Tener inicializado Git:
 - ***git init***
2. Agregar los archivos al área de preparación:
 - ***git add . (para todos los archivos)***
3. Crear un commit:
 - ***git commit -m "Descripción del cambio"***
4. Vincular el repositorio local con el remoto en GitHub (solo la primera vez)
 - ***git remote add origin <URL del repositorio>***
5. Subir commit a GitHub:
 - ***git push origin main***

h. ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión de un repositorio de Git almacenado en GitHub, en lugar de estar solo en la computadora en forma local. Este tipo de repositorio permite a varios desarrolladores colaborar en un mismo proyecto, sincronizando sus cambios a través de push (enviar cambios) y pull (obtener cambios).

i. ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git en la consola debemos ejecutar el siguiente comando: ***git clone <URL de tu repositorio remoto>***

j. ¿Cómo empujar cambios a un repositorio remoto?

Para empujar cambios del repositorio local al repositorio remoto (Git to GitHub) debemos utilizar el comando push: ***git push origin main***

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

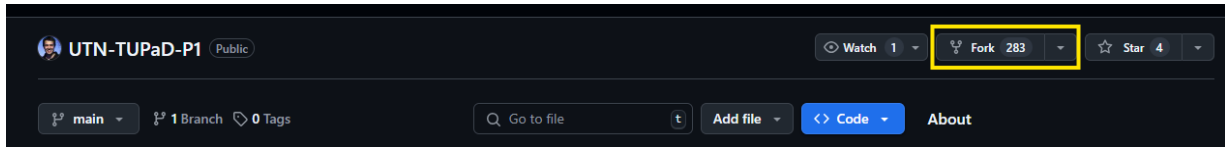
Para tirar cambios del repositorio remoto al repositorio local (GitHub to Git) debemos utilizar el comando push: ***git pull origin main***

l. ¿Qué es un fork de repositorio?

Un fork de un repositorio es básicamente una copia del repositorio de otra persona que se realiza en tu propia cuenta de GitHub. Realizar esta copia permite modificar, experimentar o proponer cambios en el código sin afectar al código o repositorio original, únicamente se realizan en tu copia.

m. ¿Cómo crear un fork de un repositorio?

Para realizar un fork debemos dirigirnos al repositorio que nos interese copiar y utilizar el botón “Fork” (arriba a la derecha), completar la información solicitada y se realizara la copia a nuestra cuenta.



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

Si hicimos cambios en nuestro fork y queremos enviarlos al repositorio original para que los creadores del repositorio los tengan en consideración debemos hacer un Pull Request (PR):

1. Subir los cambios a nuestro fork con git push.
2. En GitHub, ir a la pestaña Pull Requests del repositorio original.
3. Hacer clic en New Pull Request y seleccionar nuestro fork.
4. Enviar la solicitud para que los creadores revisen y fusionen los cambios si fuesen considerados.

o. ¿Cómo aceptar una solicitud de extracción?

Para aceptar cambios (pull request) en nuestro repositorio debemos:

1. Acceder a GitHub y dirigirnos a la sección “Pull Request”.
2. Revisar la solicitud, se puede comentar si se necesitan cambios adicionales o verificar los incluidos.
3. Aprobar haciendo clic en “Review changes” y luego en “Approve”.
4. Si no hay conflictos, podremos fusionar nuestros códigos clickeando en “Merge pull request”. En caso que haya conflictos debemos corregirlos antes.

p. ¿Qué es un etiqueta en Git?

En Git, una etiqueta (tag) es una referencia a un punto específico en el historial de commits, generalmente utilizada para marcar versiones importantes del código.

Sirven para marcar versiones estables del software, identificar puntos clave en el desarrollo y facilitar la recuperación de versiones específicas.

q. ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta (tag) en Git podemos utilizar el siguiente comando:

git tag <nombre>

r. ¿Cómo enviar una etiqueta a GitHub?

Si queremos enviar una etiqueta especifica debemos utilizar el comando:

git push origin nombre-del-tag

Y en caso de querer enviar todas las etiquetas:

git push --tags

s. ¿Qué es un historial de Git?

El historial de Git es el registro de todos los cambios realizados en un repositorio, organizado en forma de commits. Permite ver qué modificaciones se hicieron, quién las realizó y en qué momento.

t. ¿Cómo ver el historial de Git?

Para ver el historial en git podemos usar los comandos:

git log (para ver todo el historial de commits) o **git log --oneline** (para ver el historial en una línea por commit)

u. ¿Cómo buscar en el historial de Git?

Tenemos varias formas de buscar en el historial, según nuestra necesidad:

1. Por palabra clave del mensaje en el commit: **git log --grep="palabra clave"**
2. Por autor: **git log --author="nombre del autor"**
3. En un archivo específico: **git log --nombre del archivo**
4. Por palabra clave en el código: **git grep "palabra clave"**
5. Por fechas: **git log --since="2025-03-01" until="2025-03-28"**

v. ¿Cómo borrar el historial de Git?

No se recomienda borrar el historial, es una decisión drástica ya que eliminaría todos los commits del repositorio.

w. ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un repositorio que solo puede ser visto y accedido por los usuarios que el propietario invite. No es visible para el público en general y es útil para proyectos privados, desarrollo interno o código confidencial.

x. ¿Cómo crear un repositorio privado en GitHub?

1. Iniciar sesión en GitHub y hacer click en el botón "+" arriba a la derecha y seleccionar "New repository"
2. Escribir el nombre del repositorio.
3. En la sección "Visibility" seleccionar "Private"
4. Hacer click en "Create repository"

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

Para invitar a alguien a un repositorio privado debemos:

1. Abrir el repositorio en GitHub e ir a la pestaña "Settings".
2. En el menú lateral, seleccionar "Collaborators".
3. Hacer clic en "Add people" e ingresar el nombre de usuario o correo de GitHub de la persona a invitar
4. Hacer clic en "Add" y GitHub enviará la invitación.

z. ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es accesible para cualquier persona en Internet. Todos pueden ver el código, clonarlo y, si tienen permisos, contribuir. Es ideal para proyectos de código abierto.

aa. ¿Cómo crear un repositorio público en GitHub?

Se crea de la misma manera que el privado (pregunta x) pero en visibility se selecciona "Public"

bb. ¿Cómo compartir un repositorio público en GitHub?

Podemos compartir nuestro repositorio publico copiando únicamente el link URL del repositorio y compartiéndolo con quienes deseamos.

2) Realizar la siguiente actividad:

a. Crear un repositorio:

1. Dale un nombre al repositorio.
2. Elige el repositorio sea público.
3. Inicializa el repositorio con un archivo.

Create a new repository

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

Required fields are marked with an asterisk (*).

Owner * roangelelli / Repository name * Actividad1
Actividad1 is available.

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

Description (optional)

Este repositorio es para la actividad 2 del TP2 - Programación I - TUPAD

☒ 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

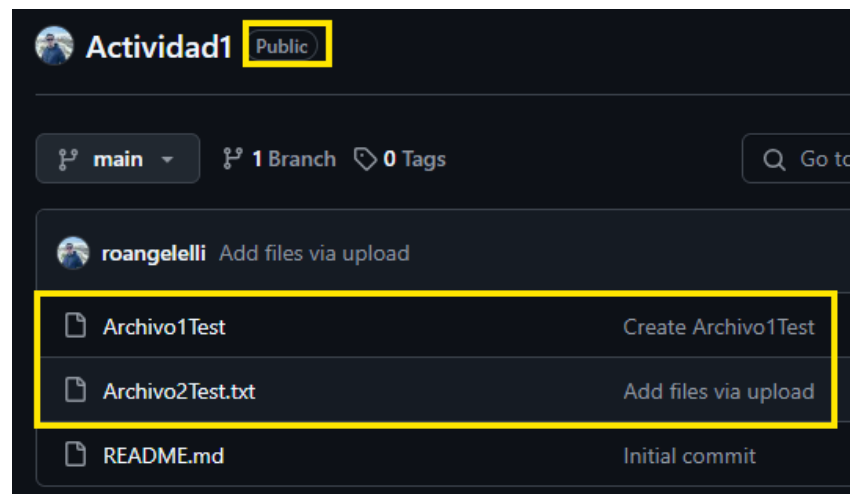
License: None

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](#).

① You are creating a public repository in your personal account.

Create repository



b. Agregando un archivo:

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

```

MINGW64/c/Users/Administrator/Desktop/Test git
Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git
$ git config --global user.email "rodrigoangelelli3@gmail.com"

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git
$ git config --global user.name "roangelelli"

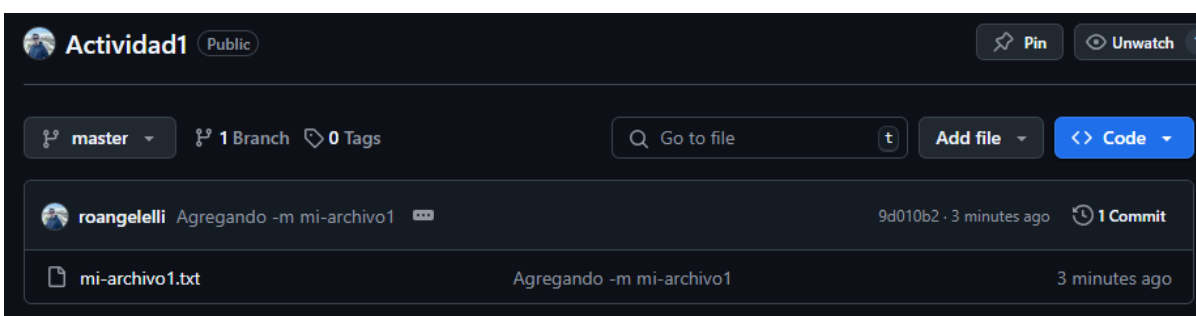
Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git
$ git init
Initialized empty Git repository in C:/Users/Administrator/Desktop/Test git/.git
/

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (master)
$ git remote add origin https://github.com/roangelelli/Actividad1.git

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (master)
$ git add .

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (master)
$ git commit -m "Agregando -m "mi-archivo1"
> git commit -m "Agregando -m mi-archivo1
[master (root-commit) 9d010b2] Agregando -m mi-archivo1 git commit -m Agregando
1 file changed, 4 insertions(+)
create mode 100644 mi-archivo1.txt

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 263 bytes | 263.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/roangelelli/Actividad1.git
* [new branch]      master -> master
  
```



a. Creando Branchs:

4. Crear una Branch.
5. Realizar cambios o agregar un archivo.
6. Subir la Branch.

```
Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (master)
$ git branch -a
* master
remotes/origin/master

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (master)
$ git branch NuevaBranch

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (master)
$ git checkout NuevaBranch
Switched to branch 'NuevaBranch'

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (NuevaBranch)
$ git add .

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (NuevaBranch)
$ git commit -m "agregando miarchivo1copia"
[NuevaBranch 855f150] agregando miarchivo1copia
1 file changed, 4 insertions(+)
create mode 100644 mi-archivo1 - copia.txt

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (NuevaBranch)
$ git push origin NuevaBranch
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 262 bytes | 262.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'NuevaBranch' on GitHub by visiting:
remote:   https://github.com/roangelelli/Actividad1/pull/new/NuevaBranch
remote:
To https://github.com/roangelelli/Actividad1.git
 * [new branch] NuevaBranch -> NuevaBranch

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (NuevaBranch)
$ |
```

The screenshot shows the GitHub web interface for the repository 'roangelelli/Actividad1'. The 'NuevaBranch' branch is selected, and it is shown to be 1 commit ahead of the 'master' branch. The commit history is visible, showing two commits: 'mi-archivo1 - copia.txt' (2 minutes ago) and 'mi-archivo1.txt' (11 minutes ago). The 'mi-archivo1 - copia.txt' commit is highlighted with a yellow box.

3) Realiza la siguiente actividad:

a. **Paso 1:** Crear un repositorio en GitHub

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

Repository name * conflict-exercise
conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about [scaling-computing-machine](#) ?

Description (optional)
Este repositorio se crea para la actividad 3, del TP2 de Programación 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.](#)

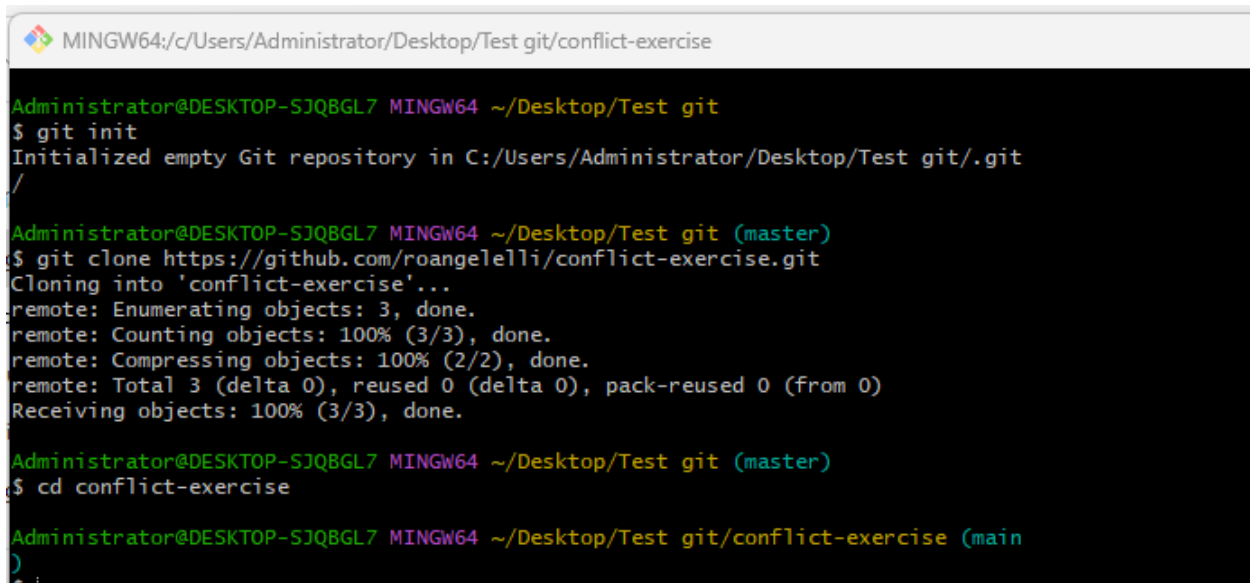
This will set `main` as the default branch. Change the default name in your [settings](#).

🔔 You are creating a public repository in your personal account.

Create repository

b. **Paso 2:** Clonar el repositorio a tu máquina local.

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



```
MINGW64/c:/Users/Administrator/Desktop/Test git/conflict-exercise

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git
$ git init
Initialized empty Git repository in C:/Users/Administrator/Desktop/Test git/.git
/

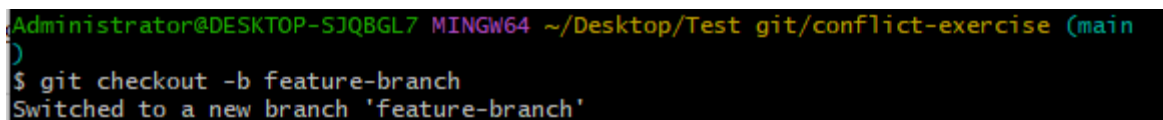
Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (master)
$ git clone https://github.com/roangelelli/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git (master)
$ cd conflict-exercise

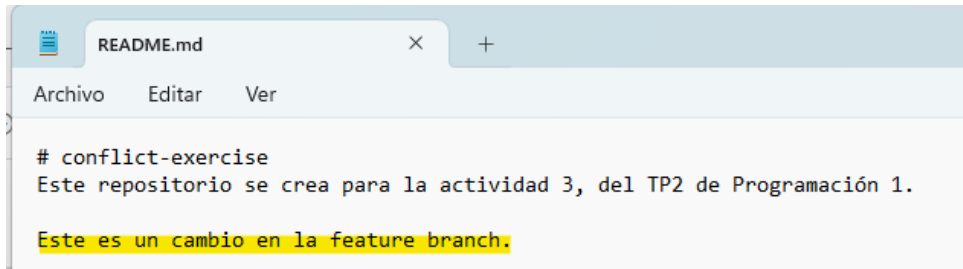
Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/conflict-exercise (main
)
```

c. **Paso 3:** Crear una nueva rama y editar un archivo

1. Crea una nueva rama llamada feature-branch:
 - `git checkout -b feature-branch`
2. 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.
3. Guarda los cambios y haz un commit:
 - `git add README.md`
 - `git commit -m "Added a line in feature-branch"`



```
Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/conflict-exercise (main
)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
```



```

# conflict-exercise
Este repositorio se crea para la actividad 3, del TP2 de Programación 1.

Este es un cambio en la feature branch.
  
```

```

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/conflict-exercise (feature-branch)
$ git add README.md

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 1280031] Added a line in feature-branch
1 file changed, 2 insertions(+)
  
```

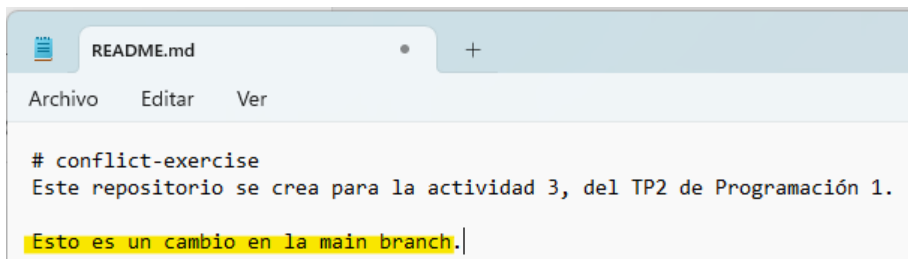
d. **Paso 4:** Volver a la rama principal y editar el mismo archivo.

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

```

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/conflict-exercise (main)
  
```



```

# conflict-exercise
Este repositorio se crea para la actividad 3, del TP2 de Programación 1.

Esto es un cambio en la main branch.
  
```

```

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/conflict-exercise (main)
$ git add README.md

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 96bf153] Added a line in main branch
1 file changed, 2 insertions(+)
  
```

e. **Paso 5:** Hacer un merge y generar un conflicto.

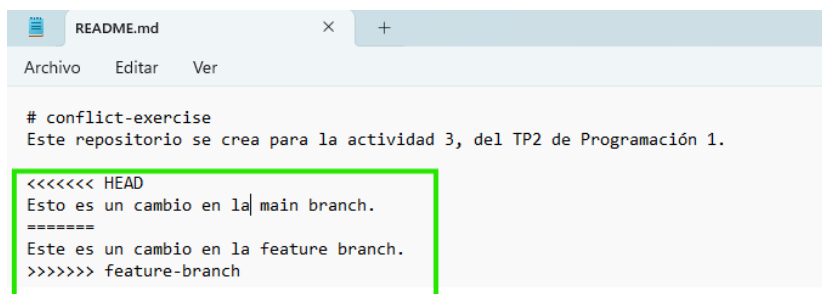
1. Intenta hacer un merge de la feature-branch en la rama main:
 - `git merge feature-branch`
2. Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md.

```
Administrator@DESKTOP-SJQBG7 MINGW64 ~/Desktop/Test git/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.
Administrator@DESKTOP-SJQBG7 MINGW64 ~/Desktop/Test git/conflict-exercise (main|MERGING)
```

f. **Paso 6:** Resolver el conflicto.

1. 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
```
2. Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
3. 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).
4. Añade el archivo resuelto y completa el merge:
 - `git add README.md`
 - `git commit -m "Resolved merge conflict"`



```
Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/conflict-exercise (main|MERGING)
$ git add README.md

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main 718041a] Resolved merge conflict

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/conflict-exercise (main)
$
```

g. **Paso 7:** Subir los cambios a GitHub.

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

```
Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/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), 790 bytes | 197.00 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/roangelelli/conflict-exercise.git
   828a16c..718041a  main -> main

Administrator@DESKTOP-SJQBGL7 MINGW64 ~/Desktop/Test git/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/roangelelli/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/roangelelli/conflict-exercise.git
   * [new branch]      feature-branch -> feature-branch
```

h. **Paso 8:** Verificar en GitHub

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



main 2 Branches 0 Tags

roangelelli Initial commit 828a16c · 18 minutes ago 4 Commits

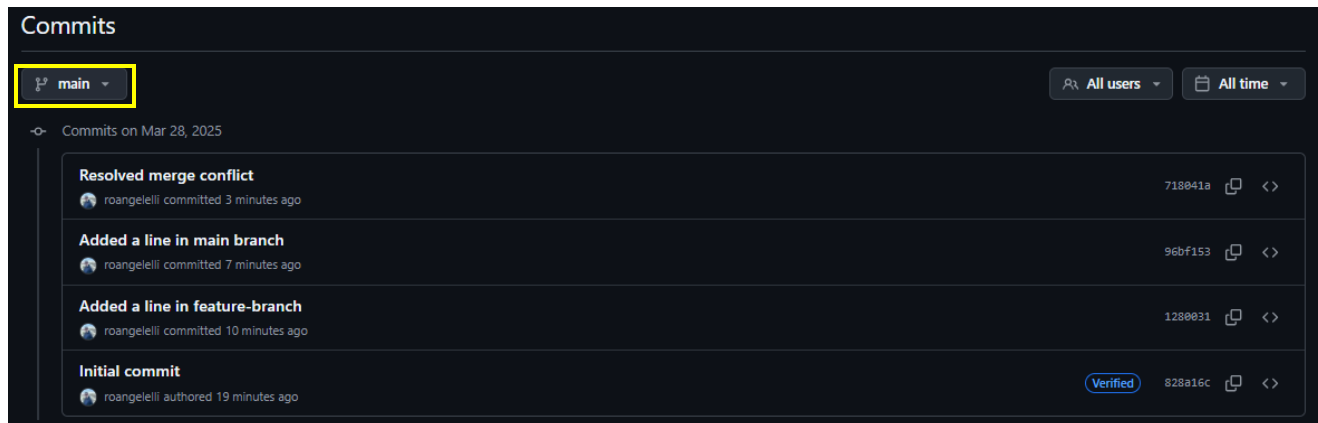
README.md Resolved merge conflict 2 minutes ago

README

conflict-exercise

Este repositorio se crea para la actividad 3, del TP2 de Programación 1.

Esto es un cambio en la main branch. Este es un cambio en la feature branch.

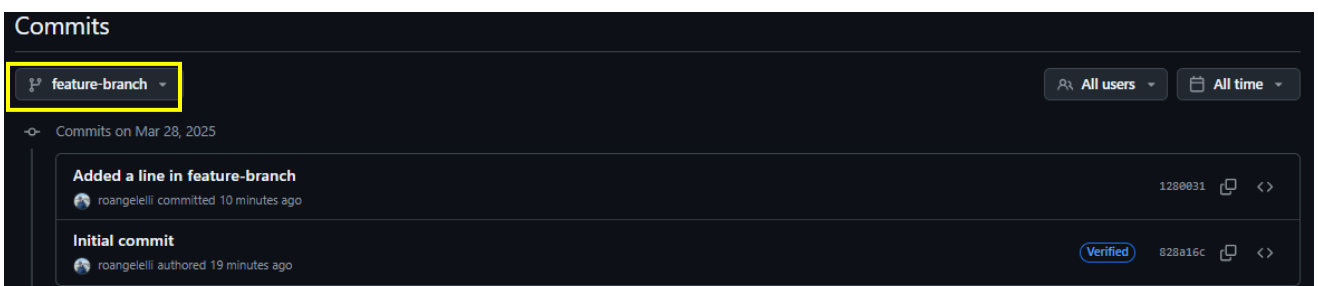


Commits

main All users All time

Commits on Mar 28, 2025

- Resolved merge conflict roangelelli committed 3 minutes ago 718041a
- Added a line in main branch roangelelli committed 7 minutes ago 96bf153
- Added a line in feature-branch roangelelli committed 10 minutes ago 1280031
- Initial commit roangelelli authored 19 minutes ago Verified 828a16c



Commits

feature-branch All users All time

Commits on Mar 28, 2025

- Added a line in feature-branch roangelelli committed 10 minutes ago 1280031
- Initial commit roangelelli authored 19 minutes ago Verified 828a16c