Práctico 2: Git y GitHub

1)

• ¿Qué es GitHub?

GitHub es como una "nube" para guardar y compartir código. Es un sitio web donde puedes almacenar proyectos, trabajar en equipo y hacer un seguimiento de los cambios en tu código usando Git.

• ¿Cómo crear un repositorio en GitHub?

1. Primero se crea una cuenta o se inicia sesión en la pagina de github.
2. Se hace click en el botón que dice new.
3. Se le pone un nombre al repositorio.
4. Se elige si es publico o privado.
5. Se marca la casilla “Add a README file” para que el repositorio tenga una descripción (opcional).
6. Se hace click en “Create repository” y listo.

• ¿Cómo crear una rama en Git?
Una rama se puede crear para tener una copia del proyecto donde se pueden hacer cambios sin afectar la versión principal. Para crear una rama se utiliza el comando “git branch nombre-de-la-rama”.

• ¿Cómo cambiar a una rama en Git?

Para cambiar de rama se utiliza el comando “git checkout nombre-de-la-rama”.

• ¿Cómo fusionar ramas en Git?

Si se hicieron cambios en una rama y se quiere combinar con la principal se utiliza los comandos:

1. Primero se cambia a la rama principal:
   git checkout main
2. Luego se combina con la otra rama:
   git merge nombre-de-la-rama

• ¿Cómo crear un commit en Git?

Un commit se hace para que se guarden los cambios hechos en el proyecto en git. Para crearlo se utilizan los siguientes comandos:

1. git add . (el . significa todos los archivos modificados)
2. git commit –m “descripción del cambio”

• ¿Cómo enviar un commit a GitHub?

Despues de hacer un commit en git, para enviarlo a github al repositorio en línea se utilizan los siguientes comandos:

1. Primero nos aseguramos de estar en la rama en la que estamos trabajando:
   git checkout main
2. Y para subir los cambios se utiliza:
   git push origin main
3. Y si es la primera vez que se sube la rama se utiliza:

git push –u origin nombre-de-la-rama

• ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión de un proyecto guardada en internet en sitios como Github. Esto permite que varias personas trabajen en el mismo código y mantengan los cambios sincronizados.

• ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Github hay que hacer lo siguiente:

1. Copiar la URL del repositorio en github.
2. Se agrega el repositorio remoto con el comando:
   git remote add origin URL_DEL_REPOSITORIO
3. Para verificar que el repositorio se agrego correctamente se utiliza el comando:
   git remote -v

• ¿Cómo empujar cambios a un repositorio remoto?

Para empujar cambios a un repositorio remoto se usa:

1. git push origin nombre-de-la-rama
2. y si es la primera vez que se sube la rama se usa:
   git push –u origin nombre-de-la-rama

• ¿Cómo tirar de cambios de un repositorio remoto?

Para traer los cambios mas recientes de la rama desde github se utiliza el comando:

git pull origin nombre-de-la-rama

• ¿Qué es un fork de repositorio?

Un fork es una copia de otro repositorio que se guarda en la cuenta propia de github. Son útiles para hacer cambios en el proyecto sin afectar el repositorio original.

• ¿Cómo crear un fork de un repositorio?

1. Primero hay que ingresar al repositorio de github que se quiere copiar.
2. Se hace click en el botón “Fork”.
3. Y github automáticamente crea una copia del repositorio en nuestra cuenta.

• ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
Para que los cambios sean añadidos a un repositorio en github con pull request se hace lo siguiente:

1. Se suben los cambios a la rama de github:
   git push origin nombre-de –la-rama
2. Se ingresa al repositorio donde se hizo el fork.
3. Se hace click en el botón “Pull request”
4. Se selecciona la rama a fusionar.
5. Se escribe un mensaje explicando los cambios y se hace click en “Create Pull Request”.
6. Después, el dueño del repositorio puede revisra y aceptar o rechazar el pull request.

• ¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción se siguen los siguientes pasos:

1. Se ingresa al repositorio de github.
2. Se hace click en la pestana Pull Requests.
3. Se selecciona la solicitud que se quiere aceptar.
4. Se revisan los cambios.
5. Se hace click en “Merge Pull Request” para fusionar los cambios con la rama principal.
6. Se confirma la fusión haciendo click en “Confirm Merge”.

• ¿Qué es una etiqueta en Git?

Una etiqueta en git se usa para señalar versiones importantes del código. Se pueden usar marcar un punto específico en el historial del repositorio.

• ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta se usa el comando:

git tag –a nombre-de-la-etiqueta –m ”Descripción de la etiqueta”

• ¿Cómo enviar una etiqueta a GitHub?

Después de crear una etiqueta en git, se puede enviar con este comando:

git push origin nombre-de-la-etiqueta

y si se quiere eviar todas las etiquetas de una vez, se usa el comando:

git push origin –tags

• ¿Qué es un historial de Git?

El historial de git es el registro de todos los cambios que se han hecho en un repositorio. Cada vez que se hace un commit , git lo guarda para ver que modificaciones se hicieron , cuando y por quien lo cual es útil para colaborar y corregir errores.

• ¿Cómo ver el historial de Git?

Para ver el historial se usa el comando:

git log

esto permitirá ver:

un ID del commit.

Nombre del autor.

Fecha del commit.

Menaje del commit.

Y si se quiere ver el historial de forma resumida se puede hacer:

git log –oneline

el cual mostrara solo el ID corto del commit y su mensaje.

• ¿Cómo buscar en el historial de Git?

Si se necesita encontrar un commit especifico se pueden usar palabras clave usando el siguiente comando:

git log --grep=”palabra clave”

o también se pueden buscar commits de un usuario especifico:

git log –autor=”nombre”

• ¿Cómo borrar el historial de Git?

Si se quiere eliminar todo el historial y empezar de nuevo, se puede hacer lo siguiente:

1. Borra la carpeta .git (esto elimina todo el historial):
   rm –rf .git
2. Inicia un nuevo repositorio:
   git init
3. Agrega y confirma los archivos nuevamente:
   git add .
   git commit –m “Nuevo inicio del repositorio”

• ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en github es un repositorio que solo lo puede ver el usuario y las personas que invite. Se puede poner privado cuando se crea el repositorio.

• ¿Cómo crear un repositorio privado en GitHub?

Para crear un repositorio privado en github se siguen los siguientes pasos:

1. Inicia sesión en github.
2. Se hace click en el botón “New”(para crear un repositorio)
3. Se escribe el nombre del repositorio.
4. Se elige la opción “Private”
5. Se hace click en “Create repository” y listo.

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

Para invitar a alguie a un repositorio privado en github se siguen los siguientes pasos:

1. Desde el repositorio que se quiere invitar, se hace click en “Settings”
2. En el menú lateral, en la sección “Manage Access”, se hace click en “Invite a collaborator”.
3. Se escribe el usuario o correo electrónico de la persona que se quiere invitar.
4. Se hace click en “Add” y se envía la invitación.

• ¿Qué es un repositorio público en GitHub?

Un repositorio publico en github es un repositorio visible para cualquier persona en github. Todos pueden verlo pero solo las personas con permiso pueden hacer cambios. Son ideales para compartir proyectos o código abierto.

• ¿Cómo crear un repositorio público en GitHub?

Para crear un repositorio publico en github se siguen los siguientes pasos:

1. Inicia sesión en github.
2. Se hace click en el botón “New” (para crear un repositorio)
3. Se escribe el nombre del repositorio.

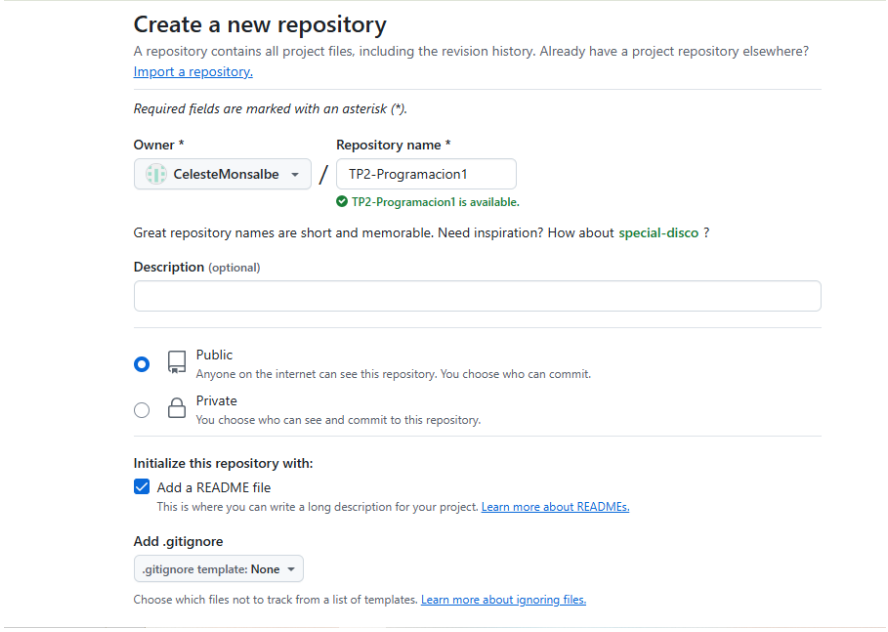
4. Se elige la opción “Public”
5. Se hace click en “Create repository” y listo.

• ¿Cómo compartir un repositorio público en GitHub?

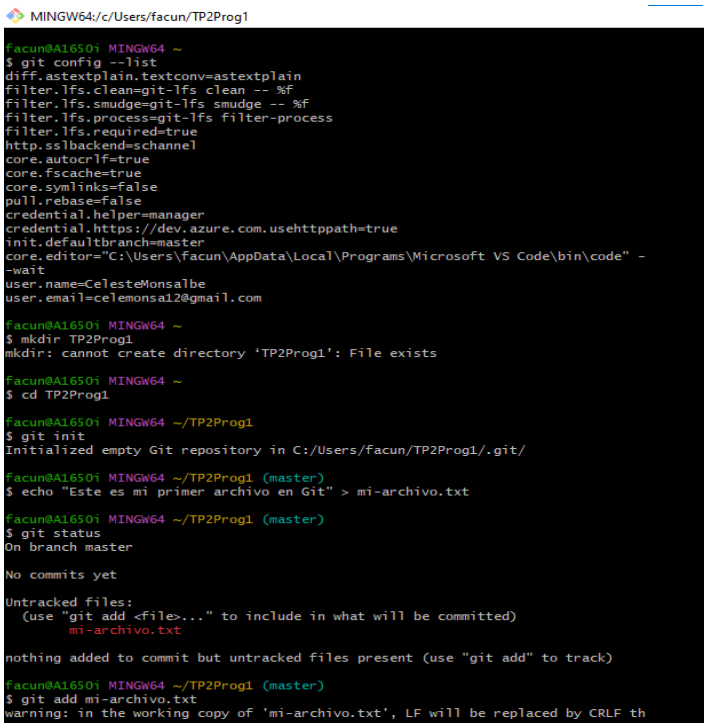
Para compartir repositorios públicos desde el repositorio de github se puede copiar la URL de la barra de direcciones.(por ejemplo: https://github.com/tu-usuario/tu-repositorio)

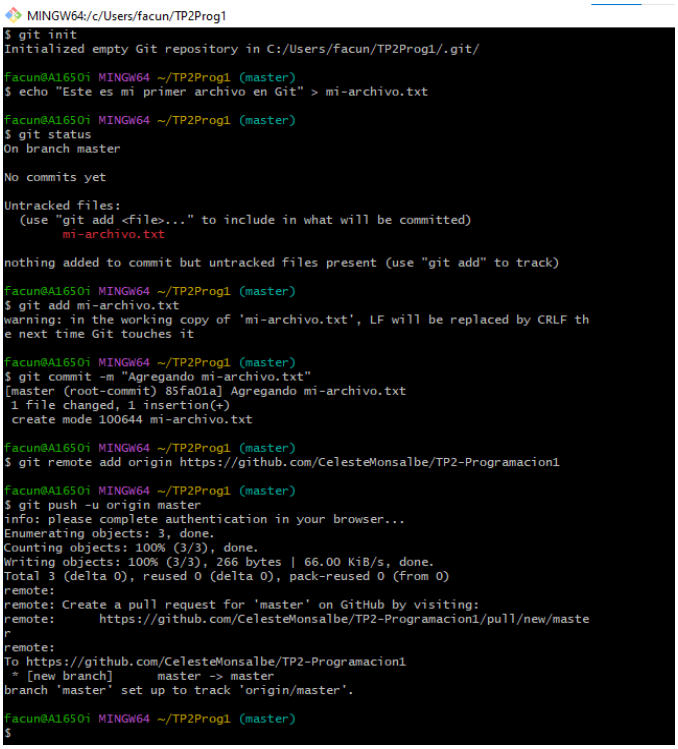
2)

Crear un repositorio:

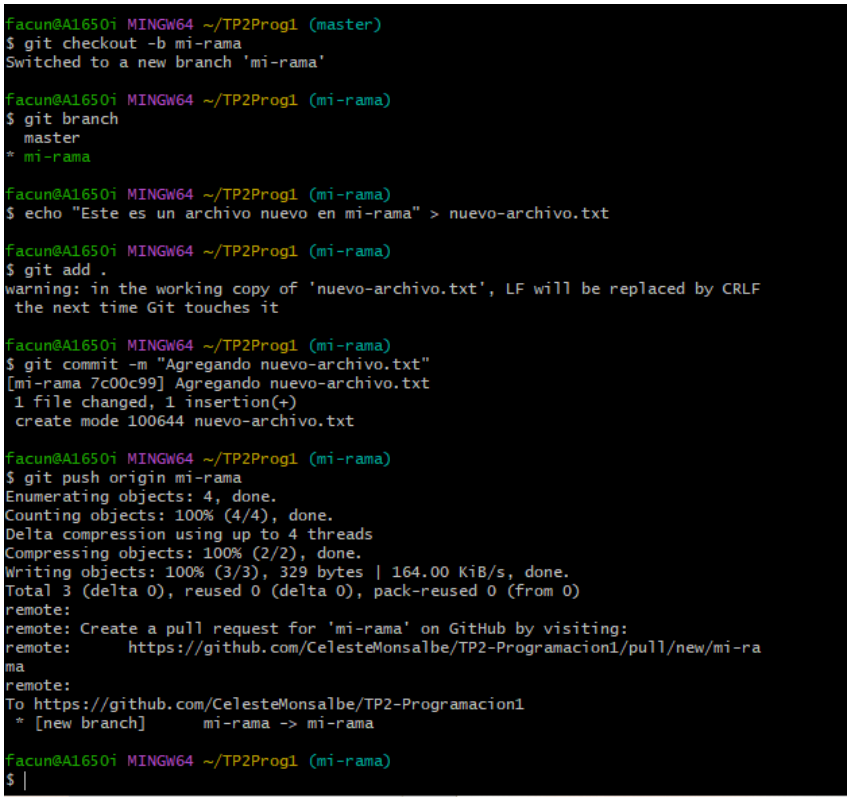


Agregando un archivo:

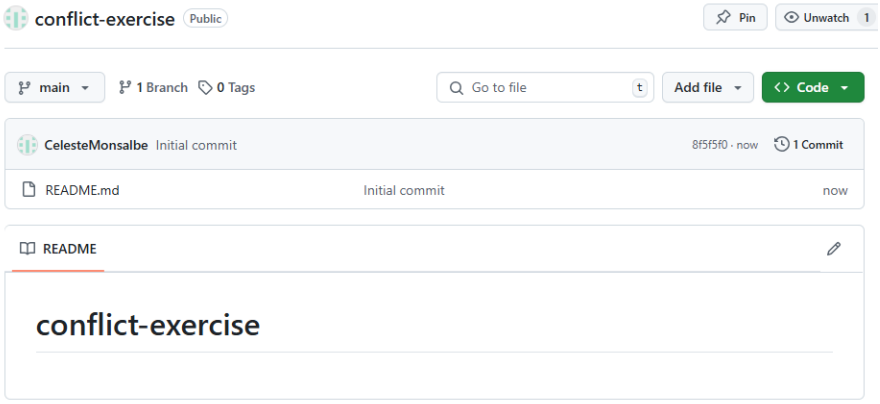




Creando branchs:



3)



MINGW64:/c/Users/facun/conflict-exercise

```
facun@A165Oi MINGW64 ~
$ git clone https://github.com/CelesteMonsalbe/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.

facun@A165Oi MINGW64 ~
$ cd conflict-exercise

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git checkout -b feature branch
fatal: 'branch' is not a commit and a branch 'feature' cannot be created from it

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

facun@A165Oi MINGW64 ~/conflict-exercise (feature-branch)
$ code README.md

facun@A165Oi MINGW64 ~/conflict-exercise (feature-branch)
$ git add README.md

facun@A165Oi MINGW64 ~/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 8993b74] Added a line in feature-branch
 1 file changed, 2 insertions(+), 1 deletion(-)

facun@A165Oi MINGW64 ~/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git add README.md

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git merge feature-branch
Updating 8f5f5f0..8993b74
Fast-forward
```

MINGW64:/c/Users/facun/conflict-exercise

```
facun@A165Oi MINGW64 ~/conflict-exercise (feature-branch)
$ git add README.md

facun@A165Oi MINGW64 ~/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch ce9fde2] Added a line in feature-branch
 1 file changed, 1 insertion(+), 1 deletion(-)

facun@A165Oi MINGW64 ~/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ code README.md

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ code README.md

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git add README.md

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git merge feature branch
merge: feature - not something we can merge

facun@A165Oi MINGW64 ~/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.

facun@A165Oi MINGW64 ~/conflict-exercise (main|MERGING)
$ code README.md

facun@A165Oi MINGW64 ~/conflict-exercise (main|MERGING)
$ code README.MD

facun@A165Oi MINGW64 ~/conflict-exercise (main|MERGING)
$ code README.md
```

MINGW64:/c/Users/facun/conflict-exercise

```
 1 file changed, 2 insertions(+), 1 deletion(-)

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ code README.md

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ code README.md

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ code README.md

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git add README.md

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 07e9646] Added a line in main branch
 1 file changed, 1 insertion(+), 1 deletion(-)

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git merge feature-branch
Already up to date.

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git checkout feature-branch
Switched to branch 'feature-branch'

facun@A165Oi MINGW64 ~/conflict-exercise (feature-branch)
$ code README.md

facun@A165Oi MINGW64 ~/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ ^C

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ code README.md

facun@A165Oi MINGW64 ~/conflict-exercise (main)
$ git checkout feature-branch
Switched to branch 'feature-branch'

facun@A165Oi MINGW64 ~/conflict-exercise (feature-branch)
$ code README.md

facun@A165Oi MINGW64 ~/conflict-exercise (feature-branch)
```

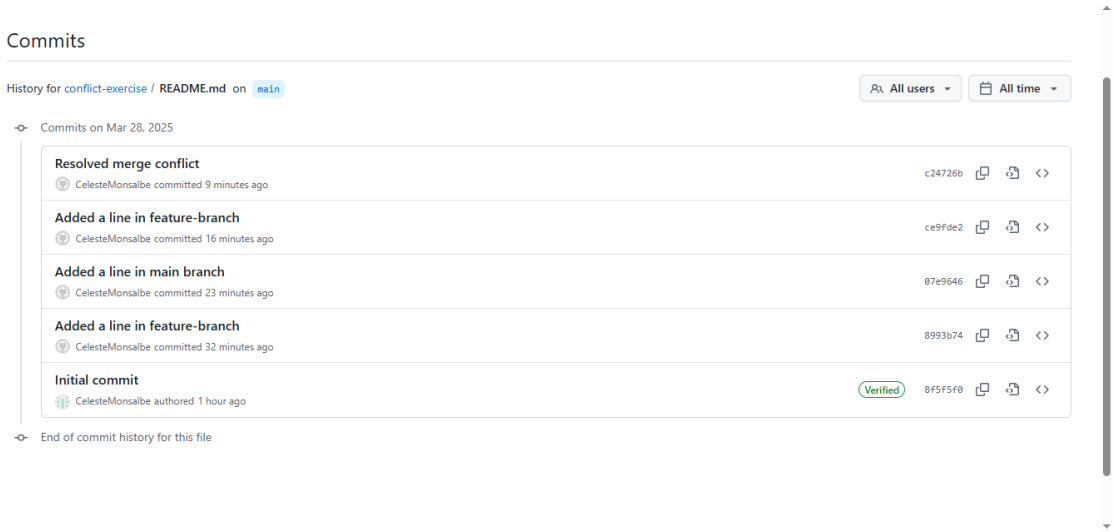
```
facun@A1650i MINGW64 ~/conflict-exercise (main|MERGING)
$ git add README.md

facun@A1650i MINGW64 ~/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main c24726b] Resolved merge conflict

facun@A1650i MINGW64 ~/conflict-exercise (main)
$ git push origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (12/12), 1.06 KiB | 154.00 KiB/s, done.
Total 12 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/CelesteMonsalbe/conflict-exercise
   8f5f5f0..c24726b  main -> main

facun@A1650i MINGW64 ~/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/CelesteMonsalbe/conflict-exercise/pull/new/featu
re-branch
remote:
To https://github.com/CelesteMonsalbe/conflict-exercise
 * [new branch]      feature-branch -> feature-branch

facun@A1650i MINGW64 ~/conflict-exercise (main)
$
```



Link repositorio actividad 2:

https://github.com/CelesteMonsalbe/TP2-Programacion1.git

Link repositorio actividad 3:

https://github.com/CelesteMonsalbe/conflict-exercise.git