

PROGRAMACIÓN I

Resolución del Trabajo Práctico N° 2: Git y Github

Leonel Agustín Serna

Ejercicio 1:

- Es una plataforma que ofrece alojamiento de repositorios de control de versiones, que permite a los desarrolladores almacenar y gestionar sus proyectos de software. Es una herramienta gratuita que permite el trabajo en equipo en proyectos, el intercambio de código y el trabajo conjunto de forma eficiente.
- Para empezar a utilizar GitHub, primero debes crear una cuenta. Puedes hacerlo en la página web de GitHub. Una vez que tengas una cuenta, puedes configurar tu perfil y empezar a crear repositorios. En esta plataforma debemos de clicar en New en la sección “Your repositories”, una vez allí colocar el nombre, una descripción opcionalmente, elegir que sea público o privado, etc. Una vez completado, damos click en Create Repository donde nos saldrán varias líneas de comando. Or create a new repository on the command line: Nos indica uno de los pasos para poder mandar nuestro repositorio local al repositorio que hemos creado en github y enlazarlo. Or push an existing repository from the command line: En este caso hace referencia a que ya tienes un repositorio local y solamente deseas mandarlo a la plataforma utilizando los siguientes comandos..
\$ git remote add origin <https://github.com/canal/nombre-repo.git>
\$ git push u- origin master
- Para crear una nueva, se utiliza el comando git branch: \$ git branch nombreRama. La rama “master” en git, es una rama definida de manera default gracias al comando git init.
- Para saltar de una rama a otra, se utiliza el comando \$ Git checkout nombre-de-rama. Se puede crear una rama y saltar directamente a ella utilizando \$ git checkout -b nuevarama.
- Para fusionar ramas en git, en la rama destino se utiliza el comando \$ Git merge ramaNueva.
- Crear un commit en Git es el proceso mediante el cual se guardan los cambios realizados en el repositorio. Un commit captura el estado actual del código en un momento dado y lo guarda en el historial. Una vez realizados los cambios se utiliza: \$ git add archivo1 (para agregar un archivo específico) o \$ git add . (Para agregar todos los archivos). Una vez realizado dicho proceso, se continua con el

comando `$ git commit -m "Mensaje de los cambios"`. Aca se describe los cambios realizados.

- Para enviar un commit a GitHub, una vez realizado el proceso de clonación del repositorio y hechos los cambios necesarios. Se empuja con el comando `$ git push origin nombreRama`
- Los repositorios remotos son versiones de tu proyecto que están alojados en internet o en cualquier otra red. Puedes tener varios de ellos, y en cada uno tendrás generalmente permisos de lectura o de lectura y escritura.
- Para vincular un repositorio local con uno remoto se utiliza el siguiente comando: `$ git remote add nombre https://github.com/usuario/repositorio.git`. Para verificar que se haya vinculado correctamente se puede utilizar el comando `$ git remote -v`
- Para empujar los cambios a un repositorio remoto se utiliza el comando: `$ git push origin nombreRama`
- Para tirar los cambios del repositorio remoto se utiliza el comando: `$ git pull origin nombreRama`
- Fork es un proceso de GitHub que nos permite tener una copia de otro repositorio donde nosotros más allá de tener la copia, es posible hacerle modificaciones.
- El primer paso es entrar en GitHub con nuestra cuenta, en la página del repositorio hacer click a la opción Fork y se generará automáticamente la copia en nuestra cuenta de GitHub. Luego queda clonarlo para poder trabajarlo desde nuestro repositorio local.
- 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. Daremos click en Create pull request donde veremos el asunto (colocamos algún mensaje global) y más abajo tenemos suficiente lugar para poder explicarnos en mencionar el porque ese cambio que hemos realizado nosotros, sería considerado como algo que al repositorio original le vendrían bien agregarlo.
- En el repositorio de Github, entrar a la pestaña "Pull Requests", selecciona la solicitud y clicar "Merge pull Request". De esta manera se sumará a su repositorio los cambios que hizo otro usuario.
- Una etiqueta es una función en Git para realizar como dice su nombre, etiquetar puntos específicos del historial como importantes. Se usa típicamente para marcar versiones de lanzamiento. (v1.0, por ejemplo).
- Hay dos tipos de etiquetas: ligeras y anotadas. Ligera es parecida a una rama que no cambia, es simplemente un puntero a un commit en específico. Las etiquetas anotadas se guardan en la base de datos de Git como objetos enteros

ya que contiene mucho más información. En cambio las ligeras se pueden utilizar para etiquetas temporales o aquellas a las que no estás interesado.

Ej: `$ git tag 1v.0`

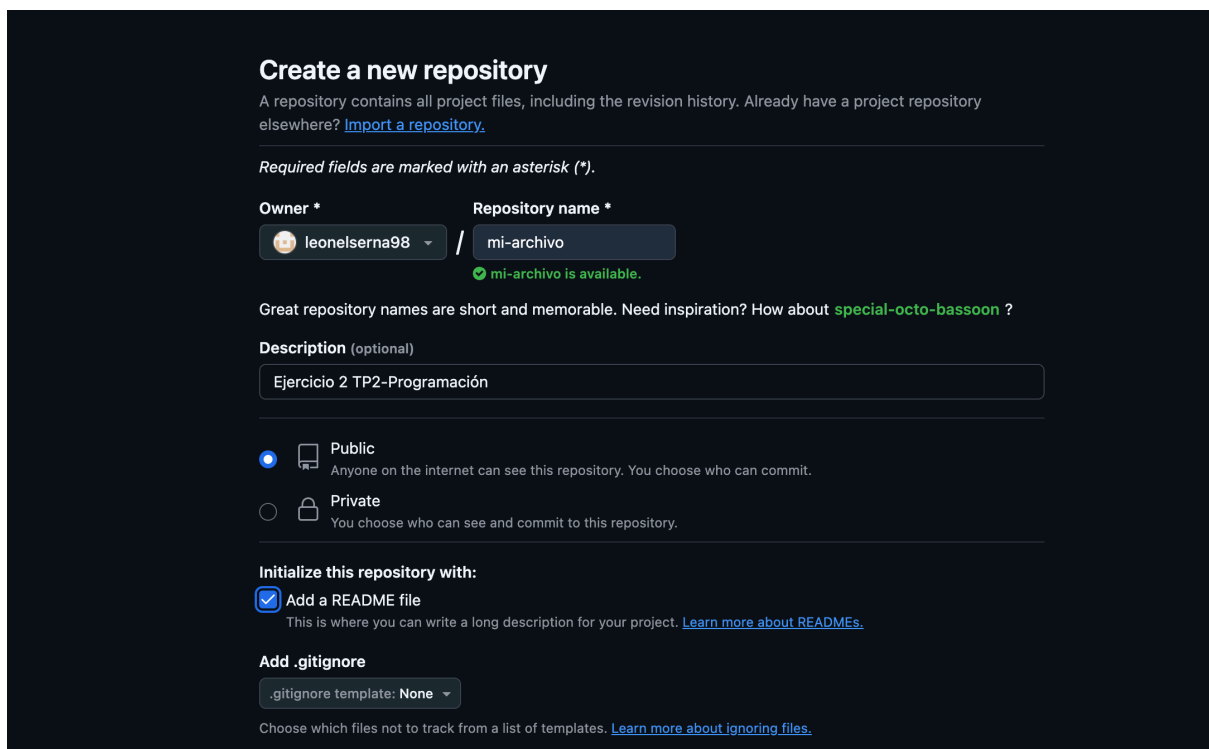
- Una vez creada la etiqueta en el repositorio local, para empujarla se utiliza el siguiente comando: `Git push origin v1.0` (ejemplo de etiqueta de respuesta anterior) o para empujar todas las etiquetas creadas, se usa: `$ git push origin --tags`.
- El historial de Git es una secuencia de todos los cambios realizados en un repositorio de Git. Cada cambio se guarda como un commit donde estos commits contienen la información sobre el estado del proyecto en un momento específico.
- Para ver el historial de Git se utiliza el comando `$ git log`. En cambio utilizando el comando `$ git log --oneline` nos muestra un resumen conciso de los commits recientes, con cada commit representado en una sola línea. Si queremos ver solo un determinado de logs se utiliza el comando `$ git log -2` (En este ejemplo muestra los últimos dos commits).
- Para buscar en el historial de commits de Git, se puede utilizar varios comandos depende como queremos filtrar. Si buscamos una palabra o frase específica se usa `$ git log --grep "palabra clave"`. Si buscamos commits con modificaciones en un archivo `$ git log --nombreArchivo`. Si buscamos en un rango de fechas `$ git log --since:"DD-MM-AAAA" --until"DD-MM-AAAA"`. Por último, si buscamos un autor en específico: `$ git log --author="Nombre del autor"`
- Para borrar el historial de Git se utiliza el comando `$ git reset`. Este comando quita del stage todos los archivos y carpetas del proyecto. Pero existen distintas formas de usarlo: `git reset nombreArchivo` (elimina archivo indicado del stage), `git reset nombreCarpeta/` (Elimina los archivos de dicha carpeta), `git reset nombrecarpeta/nombearchivo` (Quita archivo que a la vez está dentro de una carpeta), `git reset nombrecarpeta/*.extensión` (quita los archivos que cumplan con la condición indicada).
- 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.
- Iniciar sesión en GitHub, ingresa a la página de creación de repositorios, clickear `New repository` y al completar la información del repositorio elegir `Private` en la configuración de privacidad.
- En el repositorio, andar a la pestaña `Settings`, seleccionar `Collaborators` en el menú de la izquierda donde nos llevará a una página donde se puede

administrar los colaboradores Hacer click al boton “Add people” e ingres el nombre de usuario de GitHub de la persona a invitar.

- 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.
- Repetir dicho proceso para crear un repositorio privado, pero al contrario de configurarlo en Private, se debe elegir la opción Public.
- Compartiendo URL del repositorio. Se puede copiar la URL directamente desde el cuadro de texto que dice “<> Code” a la derecha en el repositorio.

Ejercicio 2:

Crear un repositorio



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 * / Repository name *

leonelserna98 / mi-archivo

mi-archivo is available.

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

Description (optional)

Ejercicio 2 TP2-Programación

☒ 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.

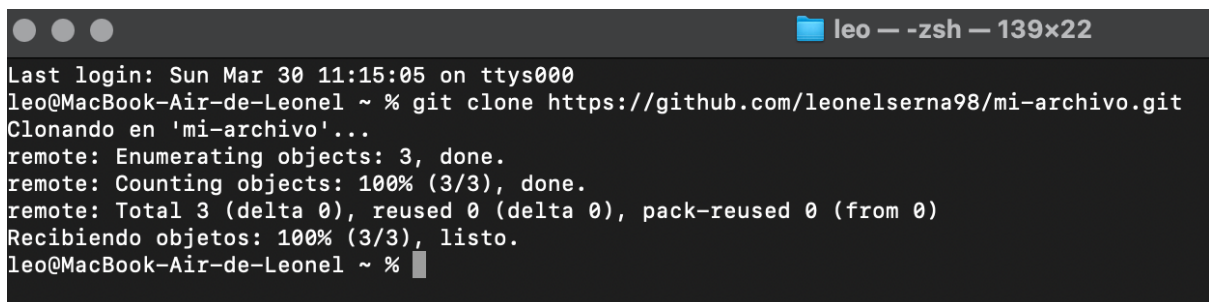
Initialize this repository with:

☒ 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.](#)



```
leo — -zsh — 139x22
Last login: Sun Mar 30 11:15:05 on ttys000
leo@MacBook-Air-de-Leonel ~ % git clone https://github.com/leonelserna98/mi-archivo.git
Clonando en 'mi-archivo'...
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)
Recibiendo objetos: 100% (3/3), listo.
leo@MacBook-Air-de-Leonel ~ %
```


Agregando un archivo

```
leo@MacBook-Air-de-Leonel ~ % cd /Users/leo/mi-archivo
```


```
leo@MacBook-Air-de-Leonel mi-archivo % echo "Este es mi archivo (Leonel)" > mi-archivo.txt
leo@MacBook-Air-de-Leonel mi-archivo %
```

```
[leo@MacBook-Air-de-Leonel mi-archivo % git add .
leo@MacBook-Air-de-Leonel mi-archivo % git commit -m "Agregando mi-archivo.txt"
[main 40b0545] Agregando mi-archivo.txt
 1 file changed, 1 insertion(+)
 create mode 100644 mi-archivo.txt
leo@MacBook-Air-de-Leonel mi-archivo %
```


mi-archivo			
Nombre	Fecha de modificación	Tamaño	Clase
> .git	hoy, 11:41	--	Carpeta
mi-archivo.txt	hoy, 11:38	28 bytes	Texto
README.md	hoy, 11:33	43 bytes	Conten...rkdown

 **mi-archivo** Public Pin Unwatch 1

main 1 Branch 0 Tags Add file Code

 **leonelserna98** Initial commit

f0ffb68 · 12 minutes ago 1 Commit

 README.md

Initial commit


12 minutes ago

README

mi-archivo


Ejercicio 2 TP2-Programación

```
leo@MacBook-Air-de-Leonel mi-archivo % git branch
* main
leo@MacBook-Air-de-Leonel mi-archivo % git push origin main
Enumerando objetos: 4, listo.
Contando objetos: 100% (4/4), listo.
Compresión delta usando hasta 8 hilos
Comprimiendo objetos: 100% (2/2), listo.
Escribiendo objetos: 100% (3/3), 329 bytes | 329.00 KiB/s, listo.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/leonelserna98/mi-archivo.git
f0fffb68..40b0545  main -> main
leo@MacBook-Air-de-Leonel mi-archivo %
```


mi-archivo
Public
Pin
Unwatch



main
1 Branch
0 Tags

Add file
Code


leonelserna98

Agregando mi-archivo.txt
40b0545 · 3 minutes ago
2 Commits

README.md	Initial commit	13 minutes ago
mi-archivo.txt	Agregando mi-archivo.txt	3 minutes ago


README


mi-archivo

Ejercicio 2 TP2-Programación


Crear banchs


```
leo@MacBook-Air-de-Leonel mi-archivo % git checkout -b nuevaRama
Cambiado a nueva rama 'nuevaRama'
leo@MacBook-Air-de-Leonel mi-archivo % echo "Modificación hecha desde nuevaRama" > mi-archivo.txt
leo@MacBook-Air-de-Leonel mi-archivo % git add .
leo@MacBook-Air-de-Leonel mi-archivo % git status
En la rama nuevaRama
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
      modificados:   mi-archivo.txt


leo@MacBook-Air-de-Leonel mi-archivo % git commit -m "Agregando modificación desde nuevaRama"
[nuevaRama 549a2f4] Agregando modificación desde nuevaRama
1 file changed, 1 insertion(+), 1 deletion(-)
leo@MacBook-Air-de-Leonel mi-archivo % git status
En la rama nuevaRama
nada para hacer commit, el árbol de trabajo está limpio
```


```
mi-archivo.txt
Modificación hecha desde nuevaRama


leo@MacBook-Air-de-Leonel mi-archivo % git push origin nuevaRama
Enumerando objetos: 5, listo.
Contando objetos: 100% (5/5), listo.
Compresión delta usando hasta 8 hilos
Comprimiendo objetos: 100% (2/2), listo.
Escribiendo objetos: 100% (3/3), 348 bytes | 348.00 KiB/s, listo.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'nuevaRama' on GitHub by visiting:
remote:   https://github.com/leonelserna98/mi-archivo/pull/new/nuevaRama
remote:
To https://github.com/leonelserna98/mi-archivo.git
 * [new branch]      nuevaRama -> nuevaRama
leo@MacBook-Air-de-Leonel mi-archivo %
```


 **mi-archivo** Public Pin Unwatch

 **nuevaRama** had recent pushes 36 seconds ago Compare & pull request


 main


 **2 Branches**


 0 Tags


Go to file 

Add file

 Code

 **leonelserna98** Agregando mi-archivo.txt 40b0545 · 8 minutes ago 2 Commits

 README.md Initial commit 18 minutes ago

 mi-archivo.txt Agregando mi-archivo.txt 8 minutes ago

Ejercicio 3:

Paso 1: Crear un repositorio en GitHub


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 *

Repository name *

 leonelserna98

/


conflict-exercise

 conflict-exercise is available.


Great repository names are short and memorable. Need inspiration? How about **sturdy-octo-waffle** ?

Description (optional)

Ejercicio 3 TP-Programación 2

☒  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.](#)

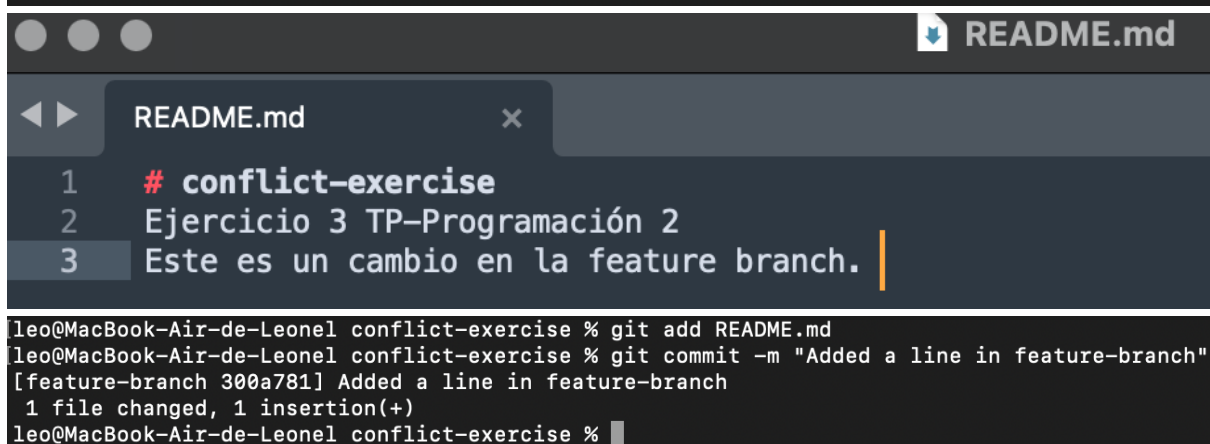
Paso 2: Clonar

```
leo — -zsh — 103x24
Last login: Sun Mar 30 11:32:11 on ttys000
leo@MacBook-Air-de-Leonel ~ % git clone https://github.com/leonelserna98/conflict-exercise.git
Clonando en '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)
Recibiendo objetos: 100% (3/3), listo.
leo@MacBook-Air-de-Leonel ~ %
```

Paso 3: Entrar en el directorio del repositorio.

Crear una nueva rama y editar un archivo


```
leo@MacBook-Air-de-Leonel ~ % cd /Users/leo/conflict-exercise
leo@MacBook-Air-de-Leonel conflict-exercise % git checkout -b feature-branch
Cambiado a nueva rama 'feature-branch'
leo@MacBook-Air-de-Leonel conflict-exercise %
```



```
leo@MacBook-Air-de-Leonel conflict-exercise % git add README.md
leo@MacBook-Air-de-Leonel conflict-exercise % git commit -m "Added a line in feature-branch"
[feature-branch 300a781] Added a line in feature-branch
1 file changed, 1 insertion(+)
leo@MacBook-Air-de-Leonel conflict-exercise %
```

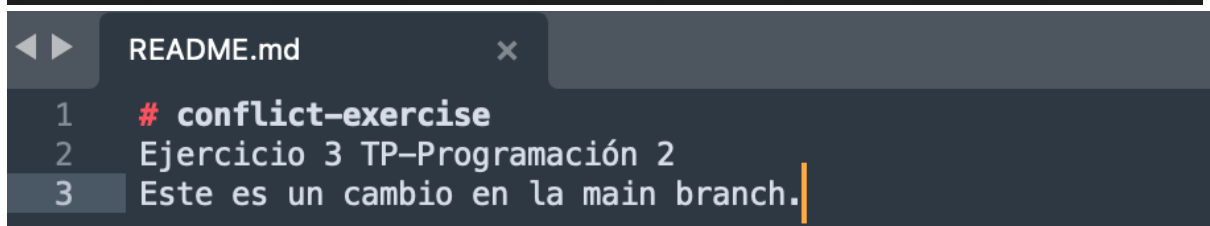
1 **# conflict-exercise**

2 Ejercicio 3 TP-Programación 2

3 Este es un cambio en la feature branch.

Paso 4: Volver a la rama principal y editar de nuevo el archivo

```
leo@MacBook-Air-de-Leonel conflict-exercise % git checkout main
Cambiado a rama 'main'
Tu rama está actualizada con 'origin/main'.
leo@MacBook-Air-de-Leonel conflict-exercise %
```



```
leo@MacBook-Air-de-Leonel conflict-exercise % git add README.md
leo@MacBook-Air-de-Leonel conflict-exercise % git commit -m "Added a line in main branch"
[main 3c5ecec] Added a line in main branch
1 file changed, 1 insertion(+)
leo@MacBook-Air-de-Leonel conflict-exercise %
```

1 **# conflict-exercise**

2 Ejercicio 3 TP-Programación 2

3 Este es un cambio en la main branch.

Paso 5: Hacer un merge y generar un conflicto

```
leo@MacBook-Air-de-Leonel conflict-exercise % git merge feature-branch
Auto-fusionando README.md
CONFLICTO (contenido): Conflicto de fusión en README.md
Fusión automática falló; arregle los conflictos y luego realice un commit con el resultado.
```

Paso 6: Resolver conflicto

```
1 # conflict-exercise
2 Ejercicio 3 TP-Programación 2
3 Este es un cambio en la main branch.
4 Este es un cambio en la feature branch.
5


leo@MacBook-Air-de-Leonel conflict-exercise % git add README.md
leo@MacBook-Air-de-Leonel conflict-exercise % git commit -m "Resolved merge conflict"
[main 77a1239] Resolved merge conflict
```

Paso 7: Subir los cambios a GitHub

```
[leo@MacBook-Air-de-Leonel conflict-exercise % git push origin main
Enumerando objetos: 11, listo.
Contando objetos: 100% (11/11), listo.
Compresión delta usando hasta 8 hilos
Comprimiendo objetos: 100% (6/6), listo.
Escribiendo objetos: 100% (9/9), 774 bytes | 774.00 KiB/s, listo.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/leonelserna98/conflict-exercise.git
a46db4b..77a1239 main -> main


leo@MacBook-Air-de-Leonel 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/leonelserna98/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/leonelserna98/conflict-exercise.git
* [new branch] feature-branch -> feature-branch
```

Paso 8: Verificar en GitHub

 **conflict-exercise** Public Pin Unwatch

main 2 Branches 0 Tags

Go to file t Add file <> Code

 **leonelserna98** Resolved merge conflict 77a1239 · 1 minute ago 4 Commits

README.md Resolved merge conflict 1 minute ago

README

conflict-exercise

Ejercicio 3 TP-Programación 2 Este es un cambio en la main branch. Este es un cambio en la feature branch.