

# Trabajo

## Practico N°2:

### Programación I

Universidad: Universidad Tecnológica Nacional

Estudiante: Tobias Leiva

Tema: Git y GitHub

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 compartís tus repositorios de forma pública o privada, también puedes almacenar y trabajar junto con otros usuarios para escribir código.

El trabajo colaborativo, una de las características fundamentales de GitHub, es posible gracias al software de código abierto Git, en el que se basa GitHub.

- ¿Cómo crear un repositorio en GitHub?

Dentro de la página de GitHub, después de haber creado una cuenta primero, en tu perfil vas haber un signo “+” y luego harás click en “New repository”.

Una vez allí, te pedirá un nombre y si quieres poner algún que otra descripción de lo que van a subir, y además de que si quieren que sea publico o privado. Luego de que este todo en orden, harás un click en Create repository y saldrán unos comandos para poner en Git

```
...or create a new repository on the command line

echo "# primer-repo1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Tobias-L7/primer-repo1.git
git push -u origin main

...or push an existing repository from the command line

git remote add origin https://github.com/Tobias-L7/primer-repo1.git
git branch -M main
git push -u origin main
```

Si todavía no has creado un repositorio local, usa el primero. Si ya tienes un repositorio local debes usar el segundo

Luego de poner los comandos anteriormente, refresca la página y ya tendrás el repositorio dentro de GitHub.

- ¿Cómo crear una rama en Git?

Antes que nada, cuando creas un repositorio, por defecto estarás en la rama “master” pero es como cualquier otra rama, no tiene nada en especial.

Para crear una nueva rama en Git deberás hacer el siguiente comando:

```
$ git branch Rama1
```

- ¿Cómo cambiar a una rama en Git?

Por defecto, estaremos en la rama “master” pero luego de haber utilizado el comando para crear otra rama podemos ir cambiando entre ramas con el siguiente comando:

```
$ git checkout Rama1
```

Y si quieres volver a la rama “master”, usas el mismo comando, pero poniendo el nombre de la rama.

También lo que se puede hacer es crear una nueva rama y saltar a ella mediante este comando:

```
$ git checkout -b Rama1
```

- ¿Cómo fusionar ramas en Git?

Para fusionar las ramas creadas en Git, primero debes posicionarte en la rama que quieres fusionar los cambios. Luego de estar en la rama, se utiliza el comando git merge para fusionar la rama creada a la rama actual.

Si yo quiero fusionar la rama “rama1” a la rama master, me posiciono en la rama master con \$ git checkout master y luego hago \$ git merge rama1 para incorporar los cambios de rama1 en master

- ¿Cómo crear un commit en Git?

Para crear un commit en Git, primero debes realizar cambios en tu repositorio. Luego de dichos cambios, harás un comando para agregar esas modificaciones que es \$ git add . o \$ git add (nombre del archivo)

Después de hacer eso, puedes hacer un commit con el comando git commit y puedes ponerle un mensaje con git commit -m “cambio en la segunda línea”

- ¿Cómo enviar un commit a GitHub?

Para poder enviar un commit a GitHub, deberás usar este comando:

```
git push origin (nombre de la rama)
```

- ¿Qué es un repositorio remoto?

Los repositorios remotos son versiones de tu proyecto que están hospedadas en Internet o en cualquier otra red. Funcionan como un punto centralizado para colaborar, almacenar, y compartir código. Permiten a múltiples desarrolladores trabajar en el mismo proyecto, sincronizando cambios y resolviendo conflictos. Son esenciales para el trabajo en equipo, la copia de seguridad del código, y la integración continua.

- ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git, debes utilizar el comando \$ git remote add para

- ¿Cómo empujar cambios a un repositorio remoto?

Para empujar cambios hechos a un repositorio remoto, debes utilizar el comando.

\$ git push origin (nombre de la rama)

- ¿Cómo tirar de cambios de un repositorio remoto?

Para tirar los cambios del repositorio remoto, hay que utilizar el comando.

\$git pull origin (nombre de la rama)

Esto es para descargar los últimos cambios del repositorio para que no haya ningún problema.

- ¿Qué es un fork de repositorio?

Muchas veces, cuando navegamos por GitHub, podemos ver repositorios de otras personas y nos parece interesante, y queremos hacerle unos cambios o modificaciones

Lo que hace fork, es que, crea una copia del repositorio y la implementa en tu cuenta de GitHub, esta copia será independientemente del repositorio de origen. Si realizas cambios en el fork, eso no afectará al repositorio principal o al original.

- ¿Cómo crear un fork de un repositorio?

Para crear un fork, debes ir al repositorio que te interesó o que quieres hacerle unas modificaciones. Luego darle click a "Fork" que está arriba a la derecha en el repositorio original.

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

Si vas a algún repositorio, verás un apartado que se llama Pull request, allí vas hacer click en new request. En esa parte, podrás escribir de porque los cambios, debido a que , etc. Y por último le dan a Create pull request.

- ¿Cómo aceptar una solicitud de extracción?

El autor del repositorio deberá clicar en Merge pull request y los cambios que acepto en esa solicitud se integran al repositorio original

- ¿Qué es una etiqueta en Git?

Git tiene la posibilidad de etiquetar puntos específicos del historial como importantes, es una referencia que señala un punto específico en el historial de un repositorio

- ¿Cómo crear una etiqueta en Git?

Se crean con el comando git tag (nombre de la etiqueta)

- ¿Cómo enviar una etiqueta a GitHub?

Primero debes crear una etiqueta en tu repositorio local, luego empujas al repositorio remoto en GitHub con el siguiente comando:

\$ git push origin (nombre de la etiqueta)

Para empujar todas las etiquetas creadas, usar:

\$git push origin --tags

- ¿Qué es un historial de Git?

Un historial de Git muestra todos los cambios realizados en un repositorio de Git-

- ¿Cómo ver el historial de Git?

Para ver el historial de Git, se deberá utilizar este comando:

\$ git log

Otra manera que se pueden visualizar con el comando \$ git log --oneline donde muestra un resumen conciso de los recientes commits representado en una sola línea, o también con el comando \$git log -5 (o puede ser otro número) que mostrará los últimos cinco commits

- ¿Cómo buscar en el historial de Git?

Para buscar en el historial de Git, hay muchos comandos que permiten filtrar y encontrar ese commits que buscas. Algunos de esos comandos son:

\$ git log --grep= (palabra o frase): este sirve para buscar palabras o frases especifica en el mensaje del commit

\$ git log --author= (nombre del autor): este sirve para buscar commits hechos por el autor

\$ git log -- (nombre del archivo): este sirve para buscar commit que han modificado un archivo especifico

- ¿Cómo borrar el historial de Git?

Para borrar el historial de Git, debes usar este comando:

\$ git reset

Pero también puedes utilizar este comando para cosas especificas usando estos:

- 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 donde solo pueden acceder allí personas autorizadas por el autor del repositorio, no se pueden clonar a diferencias de los repositorios públicos.

- ¿Cómo crear un repositorio privado en GitHub?

Cuando estas por crear un repositorio en GitHub, antes de crearlo te aparece una opción donde eliges si quieres que sea privado o público, clickeas en el privado y creas el repositorio dándole a Create repository.

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

Vas a tu repositorio privado, deberás hacer click en “Setting” del repositorio. Luego debes ir a donde dice “Collaborators”, allí encontraras una opción que dice “add people” y debes ingresar el nombre de usuario de GitHub de la persona que quieres invitar y pones como el rango que le quieres otorgar y por últimos pones add para enviar la invitación.

- ¿Qué es un repositorio público en GitHub?

Un repositorio publico en GitHub es un repositorio donde pueden acceder allí cualquier persona, sin necesidad de estar autorizados. Allí podrás clonar, hasta contribuir con el repositorio si tienen los permisos adecuados.

- ¿Cómo crear un repositorio público en GitHub?

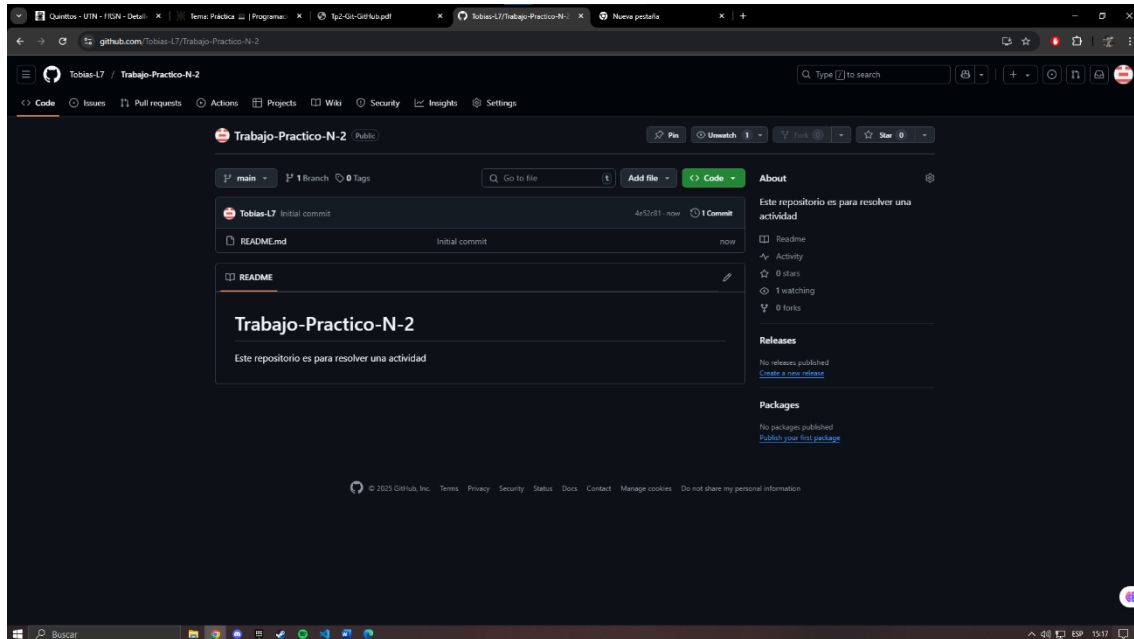
Vas a tu perfil, clickeas en el signo “+” y luego en “new repository”. Allí encontraras opciones para completar y también si quieres hacerlo privado o público, clickeas en la opción de público y sigues con el botón de “Create repository”.

- ¿Cómo compartir un repositorio público en GitHub?

Para compartir tu repositorio publico en GitHub, lo más fácil es proporcionando el enlace directo al mismo. Lo encontraras yendo a tu repositorio y en un apartado llamando “<> Code”

2) Realizar la siguiente actividad:

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



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

```
C:\Users\Clem\Trabajo-Practico-N-2>git branch
* main

C:\Users\Clem\Trabajo-Practico-N-2>git push origin main
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), 316 bytes | 316.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Tobias-L7/Trabajo-Practico-N-2.git
  4e52c81..12f6ea7  main -> main

C:\Users\Clem\Trabajo-Practico-N-2>
```

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

```
C:\Users\Clem\Trabajo-Practico-N-2>git branch rama1

C:\Users\Clem\Trabajo-Practico-N-2>git branch
* main
  rama1

C:\Users\Clem\Trabajo-Practico-N-2>git checkout rama1
Switched to branch 'rama1'

C:\Users\Clem\Trabajo-Practico-N-2>echo "Cambios hechos en rama1" > mi-archivo.txt

C:\Users\Clem\Trabajo-Practico-N-2>git add .

C:\Users\Clem\Trabajo-Practico-N-2>git commit -m "Agregando cambios en rama1"
[rama1 57bde1b] Agregando cambios en rama1
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\Clem\Trabajo-Practico-N-2>git push origin rama1
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 319 bytes | 319.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'rama1' on GitHub by visiting:
remote:   https://github.com/Tobias-L7/Trabajo-Practico-N-2/pull/new/rama1
remote:
To https://github.com/Tobias-L7/Trabajo-Practico-N-2.git
 * [new branch]      rama1 -> rama1

C:\Users\Clem\Trabajo-Practico-N-2>
```

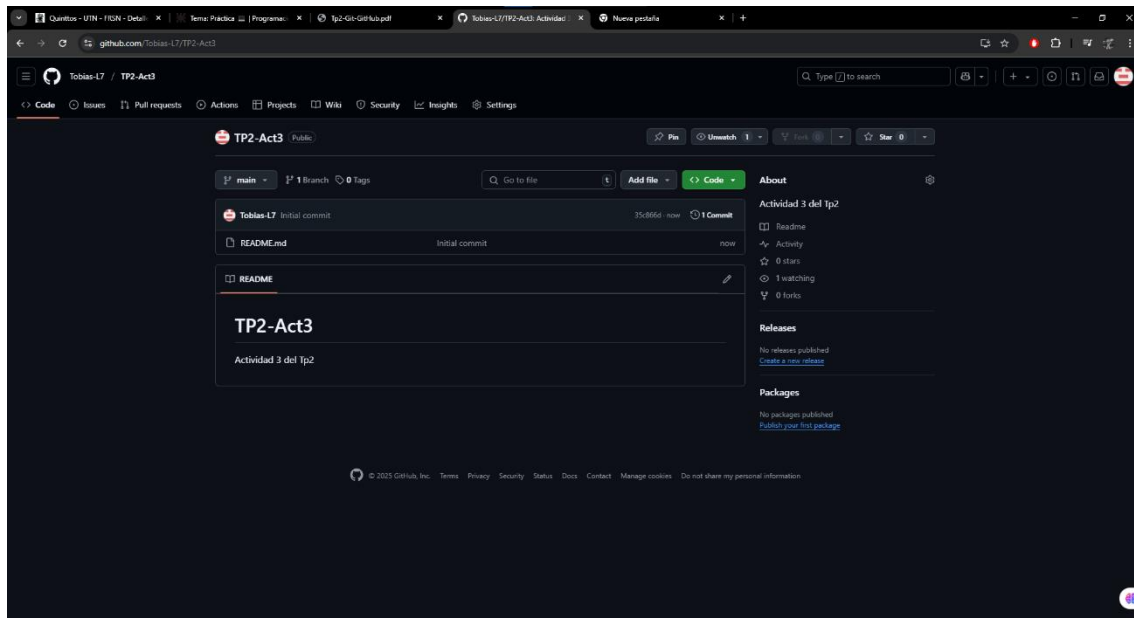
Link del repositorio Ejercicio 2: <https://github.com/Tobias-L7/Trabajo-Practico-N-2.git>



### 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".



#### 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`

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

C:\Users\Clem>git clone https://github.com/Tobias-L7/TP2-Act3.git
Cloning into 'TP2-Act3'...
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\Clem>cd TP2-Act3

C:\Users\Clem\TP2-Act3>
```

### 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:

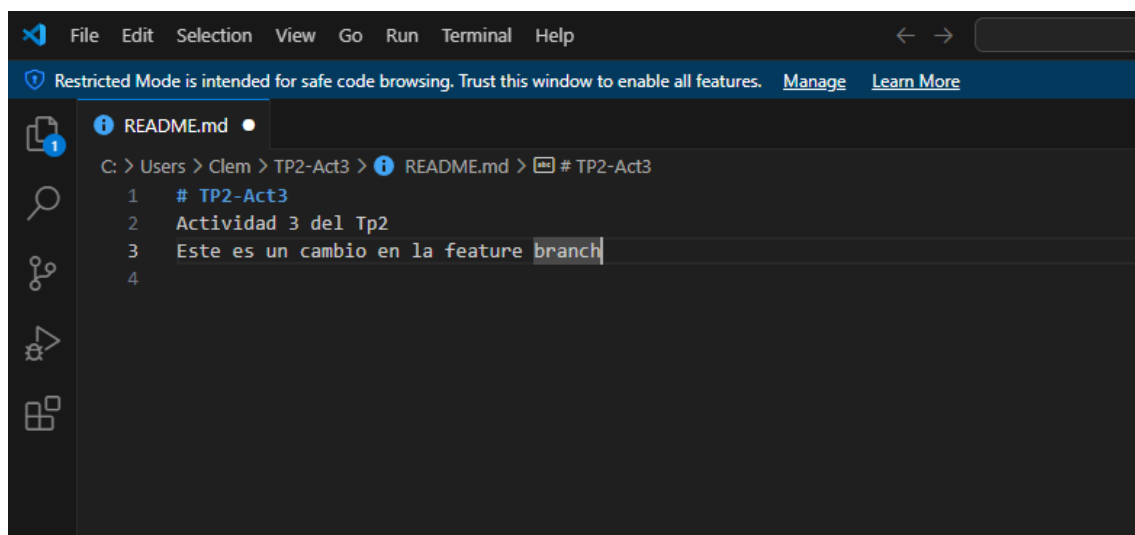
`git add README.md`

`git commit -m "Added a line in feature-branch"`

```
C:\Users\Clem\TP2-Act3>git checkout -b feature-branch
Switched to a new branch 'feature-branch'

C:\Users\Clem\TP2-Act3>git add README.md

C:\Users\Clem\TP2-Act3>git commit -m "añadimos una línea en feature-branch"
[feature-branch 9aba11c] añadimos una línea en feature-branch
1 file changed, 1 insertion(+)
```



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
```

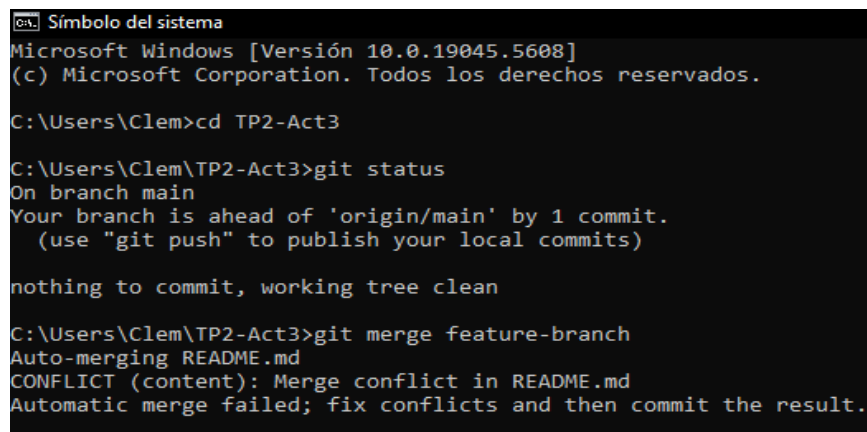
```
git commit -m "Added a line in main branch"
```

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.



```

C:\Users\Clem>cd TP2-Act3

C:\Users\Clem\TP2-Act3>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\Clem\TP2-Act3>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.
```

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.

```
=====
```

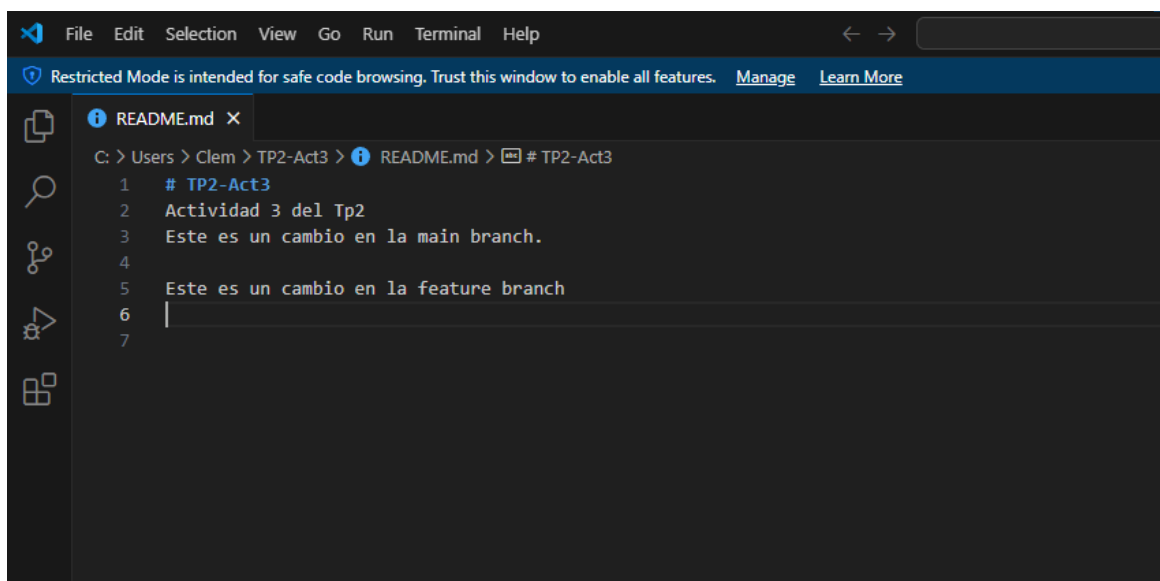
Este es un cambio en la feature branch.

```
>>>>>>> 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"
```



```
C:\Users\Clem\TP2-Act3>git add README.md
```

```
C:\Users\Clem\TP2-Act3>git commit -m "Se resolvio el conflicto"
[main 449a4f3] Se resolvio el conflicto
```

#### Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub:

```
git push origin main
```

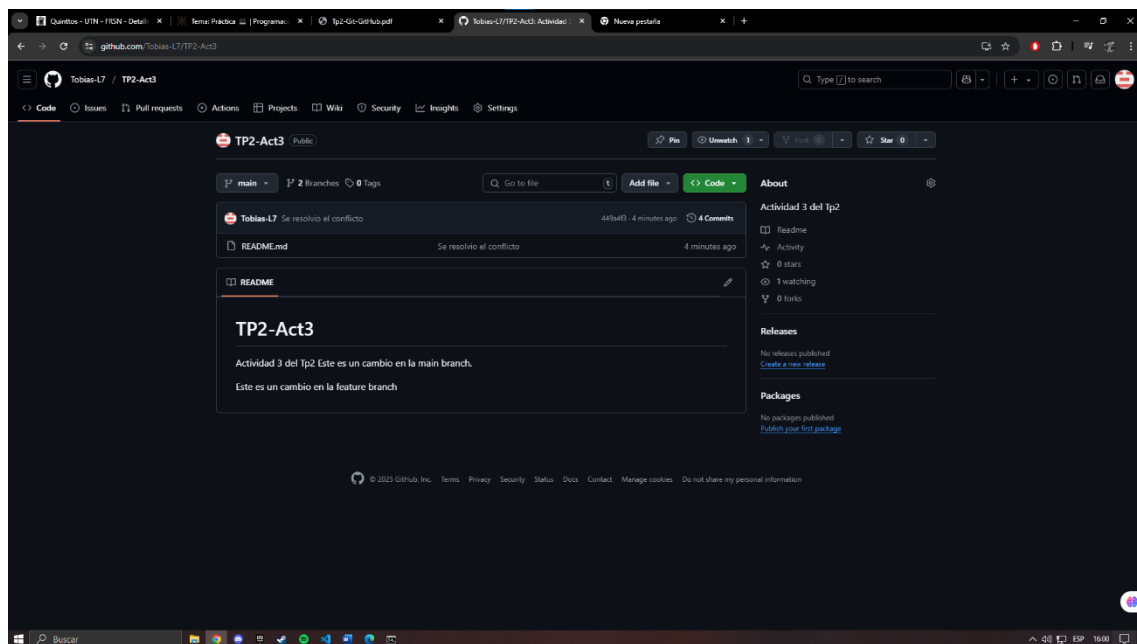
- También sube la feature-branch si deseas:

```
git push origin feature-branch
```

```
C:\Users\Clem\TP2-Act3>git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 832 bytes | 832.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Tobias-L7/TP2-Act3.git
35c866d..449a4f3 main -> main
```

### 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.



Link del repositorio Ejercicio3: <https://github.com/Tobias-L7/TP2-Act3.git>