

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

1. ¿Qué es GitHub?

GitHub es una plataforma en la nube que ofrece el servicio de alojamiento de repositorios para control de versiones, facilitando el trabajo en equipo entre desarrolladores de software. Es gratuita y de código abierto.

2. ¿Cómo crear un repositorio en GitHub?

Primero hay que tener una cuenta en GitHub. Para esto ingresamos a <https://github.com/> y hacemos click en el botón "Sign up" (Arriba a la derecha).

Una vez que tenemos la cuenta podemos crear el repositorio haciendo click en el botón verde "New". Esto nos lleva a una página donde podemos configurar el nuevo repositorio dándole un nombre, elegir si va a ser público o privado y agregar una descripción entre otras opciones. Una vez configurado el repositorio damos click en el botón "Create repository".

Ya está creado el repositorio en la nube, al cual se pueden subir archivos desde el Git local.

3. ¿Cómo crear una rama en Git?

Primero hay que posicionarse en la rama desde la que se va a crear una nueva rama, por ejemplo en la rama Main. luego ejecutar el comando:

```
$ git branch RamaNueva
```

Es importante tener en cuenta que a pesar de haber creado una rama nueva, seguimos estando en la rama Main.

4. ¿Cómo cambiar a una rama en Git?

Con el comando:

```
$ git checkout nuevaRama
```

5. ¿Cómo fusionar ramas en Git?

Para fusionar ramas en Git, primero debemos posicionarnos en la rama que recibirá los cambios de la otra, por ejemplo, la rama Main. Luego, con el comando:

```
$ git merge Rama
```

De esta manera los cambios de la rama “Rama” se incorporan a la rama “Main”

6. ¿Cómo crear un commit en Git?

Crear un *commit* es guardar, en el historial del proyecto, los últimos cambios que se han realizado.

Una vez que se ha terminado de modificar uno o más archivos de un proyecto y se quieren guardar los cambios, primero se deben agregar los mismos al área de preparación (*staging area*).

Se pueden agregar todos los archivos con:

```
$ git add .
```

O archivos específicos con:

```
$ git add archivoEjemplo.py
```

Antes de hacer el *commit* se puede revisar los cambios con:

```
$ git status
```

Para hacer el *commit* usamos el comando:

```
$ git commit -m “Mensaje breve y descriptivo de los cambios”
```

7. ¿Cómo enviar un commit a GitHub?

Para enviar los cambios realizados en un proyecto a GitHub, primero hay que crear el *commit* en el repositorio local, siguiendo los pasos del punto anterior. Luego enviamos el commit al repositorio de GitHub con:

```
$ git push origin nombreDeLaRama
```

8. ¿Qué es un repositorio remoto?

Es un repositorio que se aloja en un servidor para que puedan acceder a él distintas personas y colaborar en un proyecto en conjunto. Los colaboradores

pueden descargar el repositorio a sus computadoras, realizar cambios y, según los permisos que tengan, enviarlos al repositorio remoto.

9. ¿Cómo agregar un repositorio remoto a Git?

En el caso de que tengamos un repositorio local que queremos enviar a un remoto, lo podemos hacer con el comando:

```
$ git remote add nombre url
```

De esta manera ya podemos descargar los cambios que se hayan hecho en el remoto y/o enviar nuestro trabajo al mismo.

10. ¿Cómo empujar cambios a un repositorio remoto?

Para enviar los cambios realizados en un repositorio local al repositorio remoto, se utiliza el comando:

```
$ git push origin nombreRama
```

Es importante haber creado previamente un commit con los cambios que se quieren enviar. También hay que tener en cuenta que se podrían generar conflictos en el repositorio remoto en caso de que un colega haya estado trabajando en la misma rama, por lo que se recomienda verificar previamente el estado del código en el remoto.

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

Para actualizar el repositorio local con los cambios que se hayan realizado en el repositorio remoto, se usa el comando:

```
$ git pull origin nombreRama
```

Esto sincroniza el contenido de la rama local con la correspondiente rama remota.

12. ¿Qué es un fork de repositorio?

Un fork es una copia de un repositorio de GitHub que se crea en una cuenta personal para poder modificarlo sin afectar el repositorio original. Se utiliza cuando se quiere realizar cambios en un proyecto de código abierto sin afectar al original, se crea una versión alternativa del mismo.

13. ¿Cómo crear un fork de un repositorio?

Ingresamos al repositorio que queremos copiar en GitHub. En la parte superior derecha hacemos clic en el botón “Fork”. Esto creará una copia del repositorio en nuestra cuenta personal.

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

Un pull request es una herramienta que nos permite proponer cambios en un repositorio, es útil cuando se trabaja en colaboración o cuando se hacen modificaciones en un fork. A través del pull request, se solicita a los responsables del repositorio original que revisen y, si están de acuerdo, incorporen los cambios realizados en otra rama o en un fork.

Para enviar una pull request, primero se deben realizar los cambios en una rama diferente o en un fork del repositorio. Luego, en la página del repositorio original en GitHub, vamos a la solapa “Pull Request” y hacemos click en el botón “New pull request”.

Antes de enviar el pull request, tendremos oportunidad de comparar los cambios realizados en la rama o fork con respecto al original, y agregar una descripción de por qué estos cambios deberían ser incluidos en el mismo.

15. ¿Cómo aceptar una solicitud de extracción?

Los responsables del repositorio original pueden revisar el código de la solicitud de extracción y el mensaje descriptivo de la misma. Pueden pedir modificaciones, pero si están conformes con los cambios, pueden aceptarla haciendo clic en “Merge pull request”, y luego en “Confirm merge”.

16. ¿Qué es una etiqueta en Git?

Una etiqueta (*tag*) en Git es una referencia fija a un punto específico en la historia del proyecto, generalmente utilizada para marcar versiones o hitos importantes, por ejemplo versiones de lanzamiento de un software.

17. ¿Cómo crear una etiqueta en Git?

Se pueden crear etiquetas *lightweight* que son simplemente una referencia a un commit, o *annotated* que incluye también un mensaje y otros metadatos. Para crear una etiqueta *lightweight*:

```
git tag nombreEtiqueta
```

Para crear una etiqueta *annotated*:

```
git tag -a nombreEtiqueta -m "Mensaje de la etiqueta"
```

18. ¿Cómo enviar una etiqueta a GitHub?

Los etiquetas son referencias a commits por lo que se pueden utilizar para enviar un commit a GitHub:

```
$ git push origin nombreEtiqueta
```

19. ¿Qué es un historial de Git?

Es el registro de todos los commits realizados en un repositorio, mostrando el orden en el que se hicieron y el contenido de cada uno.

20. ¿Cómo ver el historial de Git?

Para visualizar el historial se utiliza el comando:

```
$ git log
```

O, para ver una versión resumida del historial:

```
$ git log --oneline
```

Si el proyecto es muy grande y queremos ver únicamente los últimos commits:

```
$ git log -N
```

Reemplazar "N" por el número de commits que queremos ver.

21. ¿Cómo buscar en el historial de Git?

Para buscar un término dentro del historial de commits se puede usar:

```
$ git log --grep="término"
```

Esto mostrará los commits cuyo mensaje contenga la palabra o frase buscada. El comando log también tiene opciones para buscar por fechas, o modificaciones a archivos específicos.

22. ¿Cómo borrar el historial de Git?

No se recomienda borrar el historial de un proyecto, pero si es necesario reiniciar el historial, se puede crear un nuevo repositorio o sobrescribirlo con:

```
$ rm -rf .git  
$ git init
```

También se puede utilizar:

```
$ git reset
```

Para eliminar los archivos del área de preparación (*stage*). Este comando tiene distintas opciones para eliminar archivos específicos, es importante entenderlo bien antes de usarlo, para no perder el trabajo.

23. ¿Qué es un repositorio privado en GitHub?

Es un repositorio que no es accesible públicamente. Solo los usuarios con permisos pueden ver o colaborar en él, a diferencia de los repositorios públicos que cualquier usuario puede clonar o hacerles un *fork*.

24. ¿Cómo crear un repositorio privado en GitHub?

Al crear un nuevo repositorio, se debe seleccionar la opción "*Private*" en la sección de privacidad antes de hacer clic en "*Create repository*".

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

Dentro del repositorio, ir a la pestaña "Settings", luego ingresar en "Collaborators", hacer clic en "Add people", escribir el nombre de usuario y confirmar la invitación. Luego, seleccionar el nivel de acceso del colaborador.

26. ¿Qué es un repositorio público en GitHub?

Es un repositorio que puede ser visto por cualquier persona. Cualquiera puede clonar el repositorio, ver su contenido o hacerle un *fork*.

27. ¿Cómo crear un repositorio público en GitHub?

Al crear un nuevo repositorio, se debe seleccionar la opción "*Public*" en la sección de privacidad antes de hacer clic en "*Create repository*".

28. ¿Cómo compartir un repositorio público en GitHub?

Se puede compartir la url del repositorio.

2) Realizar la siguiente actividad:

1. Crear un repositorio.

- a. Dale un nombre al repositorio.
- b. Elige que el repositorio sea público.
- c. Inicializa el repositorio con un archivo.

New 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 * Repository name *

phidalg / TP2

TP2 is available.

Great repository names are short and memorable. Need inspiration? How about fuzzy-funicular ?

Description (optional)

Ejercicio del segundo trabajo práctico de Programación I

☒ 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

2. 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.

```
MINGW64:/c/Users/pedro/OneDrive/Escritorio/TP2

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio
$ mkdir TP2

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio
$ cd TP2

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2
$ git init
Initialized empty Git repository in C:/Users/pedro/OneDrive/Escritorio/TP2/.git/

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (master)
$ echo "Un simple archivo" > archivo.txt

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (master)
$ git add .
warning: in the working copy of 'archivo.txt', LF will be replaced by CRLF the next time Git touches it

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (master)
$ git commit -m "Se agrega un simple archivo"
[master (root-commit) 158615f] Se agrega un simple archivo
1 file changed, 1 insertion(+)
create mode 100644 archivo.txt

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (master)
$ |
```

- ```
MINGW64~/c/Users/pedro/OneDrive/Escritorio/TP2
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio
$ mkdir TP2
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio
$ cd TP2
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2
$ git init
Initialized empty Git repository in C:/Users/pedro/OneDrive/Escritorio/TP2/.git/
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (master)
$ echo "Un simple archivo" > archivo.txt
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (master)
$ git add .
warning: in the working copy of 'archivo.txt', LF will be replaced by CRLF the next time Git touches it
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (master)
$ git commit -m "Se agrega un simple archivo"
[master (root-commit) f89d5ab] Se agrega un simple archivo
1 file changed, 1 insertion(+)
create mode 100644 archivo.txt
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (master)
$ git remote add origin https://github.com/phidalg/TP2.git
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (master)
$ git pull origin main --allow-unrelated-histories
```

A screenshot of a Windows terminal window titled "MINGW64:/c/Users/pedro/OneDrive/Escritorio/TP2". The terminal displays the command "Merge remoto con el archivo local|" in yellow text. Below the command, there are several blue tilde (~) symbols representing file names or paths. At the bottom of the terminal, a status bar shows ".git/MERGE\_MSG[+] [unix] (15:00 15/04/2025)" on the left and "1,34 All" on the right. The terminal background is black, and the text is primarily white and yellow.



```
MINGW64:/c:/Users/pedro/OneDrive/Escritorio/TP2

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)
Unpacking objects: 100% (3/3), 908 bytes | 50.00 KiB/s, done.
From https://github.com/phidalg/TP2
* branch main -> FETCH_HEAD
* [new branch] main -> origin/main
Merge made by the 'ort' strategy.
 README.md | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 README.md

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (master)
$ |
```

phidalg / TP2

Search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

TP2 Public

Pin Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file Add file Code

phidalg Merge remoto con el archivo local 41b7a08 · 21 minutes ago 3 Commits

README.md Initial commit 1 hour ago

archivo.txt Se agrega un simple archivo 24 minutes ago

README

# TP2

Ejercicio del segundo trabajo práctico de Programación I

About

Ejercicio del segundo trabajo práctico de Programación I

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

© 2025 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

### 3. Creando Branchs

#### a. Crear una Branch

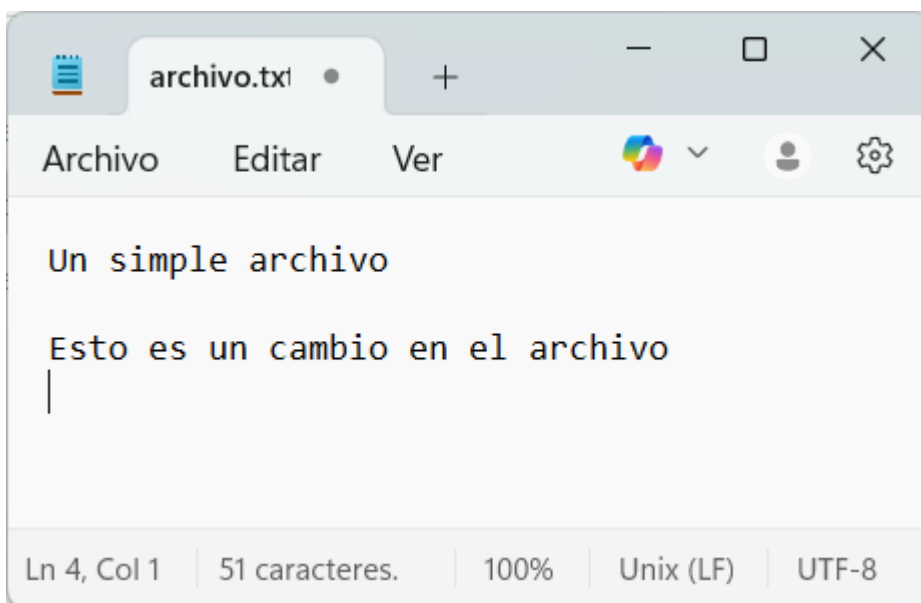
```
MINGW64:/c/Users/pedro/OneDrive/Escritorio/TP2
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (main)
$ git branch
* main

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (main)
$ git branch nuevaRama

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (main)
$ git branch
* main
 nuevaRama

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (main)
$
```

#### b. Realizar cambios o agregar un archivo



archivo.txt

Archivo Editar Ver

Un simple archivo

Esto es un cambio en el archivo

Ln 4, Col 1 | 51 caracteres. | 100% | Unix (LF) | UTF-8

#### c. Subir la Branch

```
MINGW64:/c/Users/pedro/OneDrive/Escritorio/TP2
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (nuevaRama)
$ git add .
warning: in the working copy of 'archivo.txt', LF will be replaced by CRLF the next time Git touches it

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (nuevaRama)
$ git status
On branch nuevaRama
Changes to be committed:
 (use "git restore --staged <file>..." to unstage)
 modified: archivo.txt

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (nuevaRama)
$ git commit -m "Se modifica archivo.txt"
[nuevaRama fa9b38b] Se modifica archivo.txt
1 file changed, 2 insertions(+)

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (nuevaRama)
$
```

```
MINGW64:/c:/Users/pedro/OneDrive/Escritorio/TP2
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (nuevaRama)
$ git push origin nuevaRama
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 341 bytes | 170.00 KiB/s, done.
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/phidalg/TP2/pull/new/nuevaRama
remote:
To https://github.com/phidalg/TP2.git
 * [new branch] nuevaRama -> nuevaRama

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/TP2 (nuevaRama)
$
```

Branches New branch

Overview **Yours** Active Stale All

Q Search branches...

Default

| Branch      | Updated        | Check status | Behind / Ahead | Pull request |
|-------------|----------------|--------------|----------------|--------------|
| <b>main</b> | 30 minutes ago |              | Default        | ...          |

**Your main branch isn't protected**  
Protect this branch from force pushing or deletion, or require status checks before merging. [Learn more](#) Dismiss Protect this branch

Your branches

| Branch           | Updated       | Check status | Behind / Ahead | Pull request |
|------------------|---------------|--------------|----------------|--------------|
| <b>nuevaRama</b> | 9 minutes ago |              | 0   1          | #1  ...      |

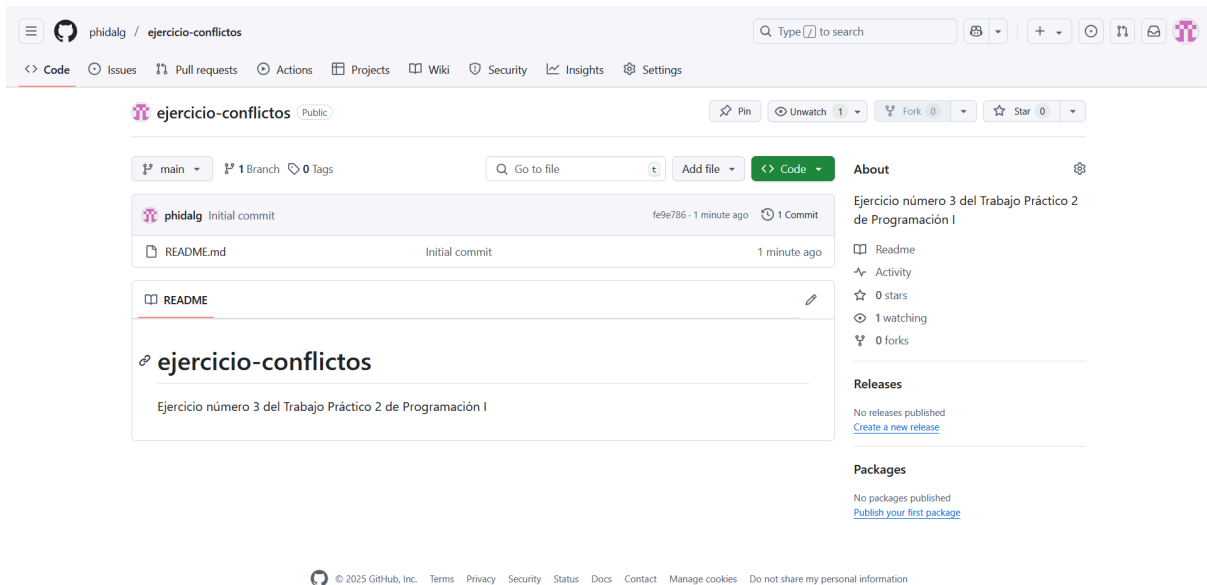
Active branches

| Branch           | Updated       | Check status | Behind / Ahead | Pull request |
|------------------|---------------|--------------|----------------|--------------|
| <b>nuevaRama</b> | 9 minutes ago |              | 0   1          | #1  ...      |

### 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
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando `git clone`
- Entra en el directorio del repositorio

```
MINGW64:/c/Users/pedro/OneDrive/Escritorio/ejercicio-conflictos

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio
$ git clone https://github.com/phidalg/ejercicio-conflictos
Cloning into 'ejercicio-conflictos'...
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.

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio
$ cd ejercicio-conflictos

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main)
$ |
```

## Paso 3) Crear una nueva rama y editar un archivo

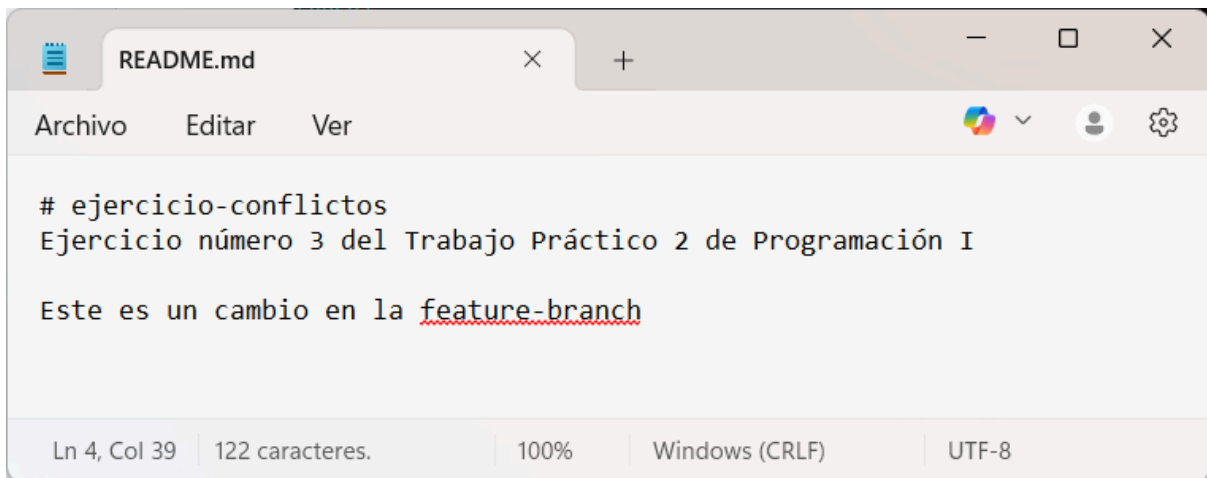
- Creas una nueva rama llamada `feature-branch`

```
MINGW64:/c/Users/pedro/OneDrive/Escritorio/ejercicio-conflictos

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (feature-b
branch)
$ |
```

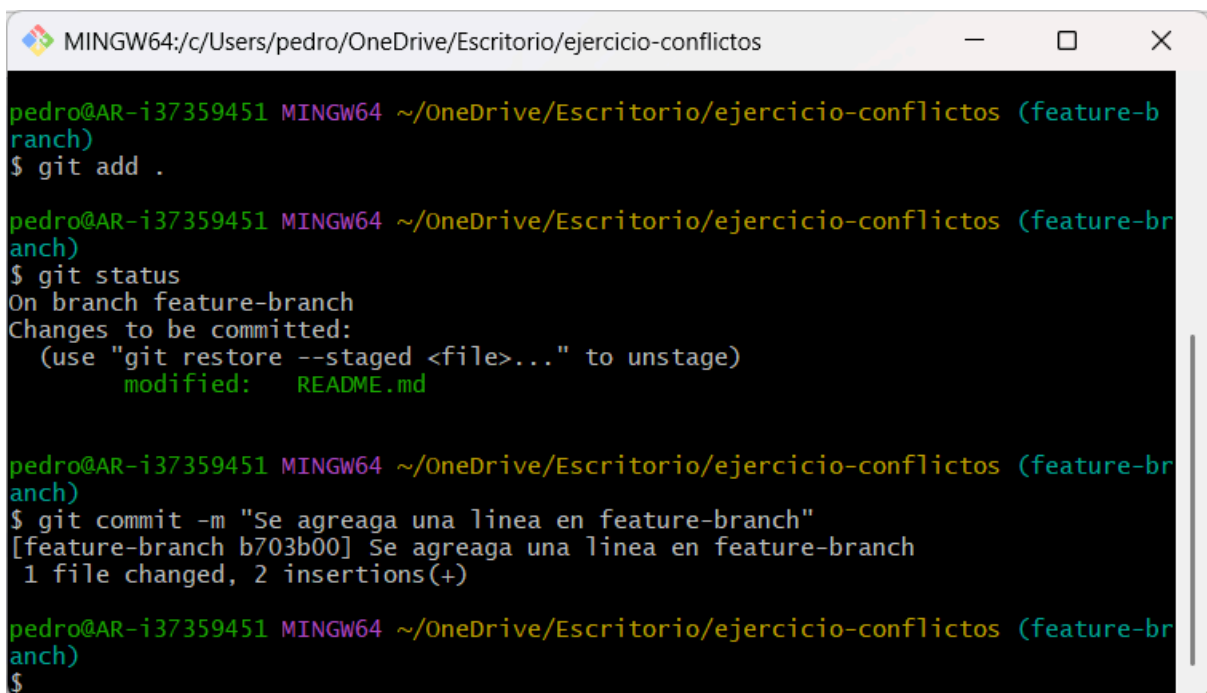
- b) Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo: Este es un cambio en la feature branch.



```
ejercicio-conflictos
Ejercicio número 3 del Trabajo Práctico 2 de Programación I

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

- c) Guarda los cambios y haz un commit:



```
MINGW64:/c/Users/pedro/OneDrive/Escritorio/ejercicio-conflictos

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (feature-b
branch)
$ git add .

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (feature-br
anch)
$ git status
On branch feature-branch
Changes to be committed:
 (use "git restore --staged <file>..." to unstage)
 modified: README.md

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (feature-br
anch)
$ git commit -m "Se agreaga una linea en feature-branch"
[feature-branch b703b00] Se agreaga una linea en feature-branch
1 file changed, 2 insertions(+)

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (feature-br
anch)
$
```

**Paso 4) Volver a la rama principal y editar el mismo archivo**

- Cambia de vuelta a la rama principal (main)
- Edita el archivo README.md de nuevo, añadiendo una línea diferente: Este es un cambio en la main branch.

```
MINGW64:/c/Users/pedro/OneDrive/Escritorio/ejercicio-conflictos

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main)
$ echo "Este es un cambio en la main branch" >> README.md

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main)
$
```

**c) Guarda los cambios y haz un commit**

```
MINGW64:/c/Users/pedro/OneDrive/Escritorio/ejercicio-conflictos

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main)
$ git add .
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time
Git touches it

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
 (use "git restore --staged <file>..." to unstage)
 modified: README.md

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main)
$ git commit -m "Agregue una linea en la rama main"
[main c83d68a] Agregue una linea en la rama main
1 file changed, 1 insertion(+)

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main)
$
```

**Paso 5) Hacer un merge y generar un conflicto**

- a) Intenta hacer un merge de la feature-branch en la rama main
- b) Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md

```
MINGW64:/c/Users/pedro/OneDrive/Escritorio/ejercicio-conflictos

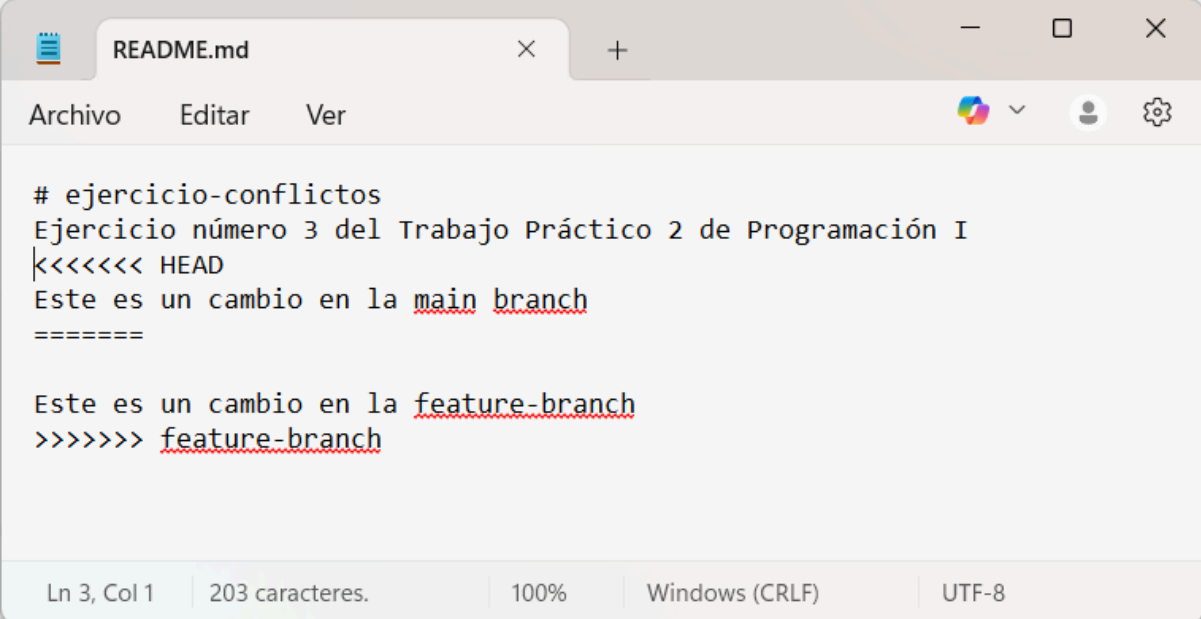
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main)
$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main|MERGING)
$ |
```

**Paso 6) Resolver el conflicto**

- a) 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
```

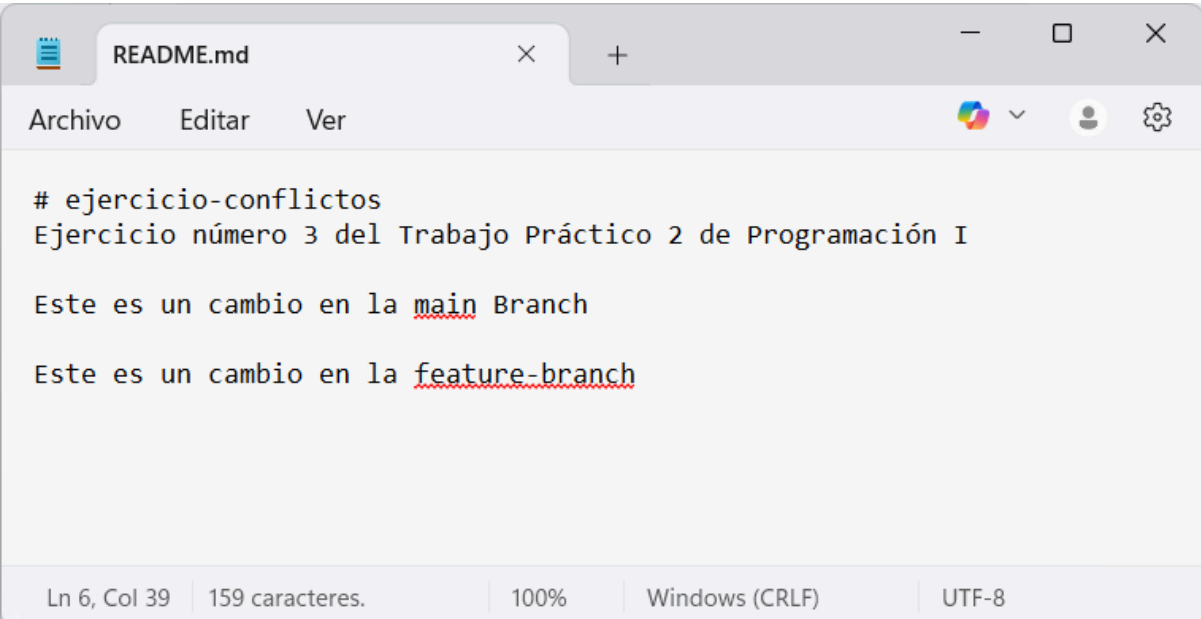


```
ejercicio-conflictos
Ejercicio número 3 del Trabajo Práctico 2 de Programación I
<<<<<<< HEAD
Este es un cambio en la main branch
=====

Este es un cambio en la feature-branch
>>>>>>> feature-branch
```

Ln 3, Col 1 | 203 caracteres. | 100% | Windows (CRLF) | UTF-8

- b) Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
- c) Edita el archivo para resolver el conflicto y guarda los cambios.



```
ejercicio-conflictos
Ejercicio número 3 del Trabajo Práctico 2 de Programación I

Este es un cambio en la main Branch

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

Ln 6, Col 39 | 159 caracteres. | 100% | Windows (CRLF) | UTF-8

**d) Añade el archivo resuelto y completa el merge**

```
MINGW64:/c/Users/pedro/OneDrive/Escritorio/ejercicio-conflictos
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main|MERGING)
$ git add .

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main|MERGING)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
 (use "git push" to publish your local commits)

All conflicts fixed but you are still merging.
 (use "git commit" to conclude merge)

Changes to be committed:
 modified: README.md

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main|MERGING)
$ git commit -m "Merge conflict resuelto"
[main 0d39352] Merge conflict resuelto

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main)
$ |
```

**Paso 7) Subir los cambios a GitHub**

- a) Sube los cambios de la rama main al repositorio remoto en GitHub
- b) También sube la feature-branch si deseas

```
MINGW64:/c/Users/pedro/OneDrive/Escritorio/ejercicio-conflictos
pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 862 bytes | 143.00 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/phidalg/ejercicio-conflictos
 fe9e786..0d39352 main -> main

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (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/phidalg/ejercicio-conflictos/pull/new/feature-branch
remote:
To https://github.com/phidalg/ejercicio-conflictos
 * [new branch] feature-branch -> feature-branch

pedro@AR-i37359451 MINGW64 ~/OneDrive/Escritorio/ejercicio-conflictos (main)
$ |
```

**Paso 8) Verificar en GitHub**

- a) Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.



