

ALUMNA: GAMARRA JOANA

Práctico 2: Git y GitHub

Actividades

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

1. ¿Qué es GitHub?


GitHub es una plataforma creada para alojar el código de las aplicaciones de los desarrolladores que la utilizan. Tiene un sistema de control de versiones y colaboración en donde los desarrolladores pueden administrar sus proyectos de software y nos permite trabajar en colaboración con otras personas desde cualquier lugar.

2. ¿Cómo crear un repositorio en GitHub?

Para crear un nuevo repositorio en GitHub lo primero que hay que hacer es iniciar sesión / crear una cuenta.

Una vez iniciada la sesión se debe oprimir el botón New, al hacerlo nos va a llevar a crear el nuevo repositorio donde debemos llenar una serie de campos que tienen que ver con el nombre, descripción de lo que se va a subir, si queremos que sea público o privado, etc.


Dueño * Nombre del repositorio *


 Joygamarra09 /

Los buenos nombres de repositorios son cortos y fáciles de recordar. ¿Necesitas inspiración? ¿Qué te parece?

[carnaval de risas ?](#)

Descripción (opcional)

☒  Público
Cualquier persona en internet puede ver este repositorio. Tú decides quién puede contribuir.

☐  Privado
Tú eliges quién puede ver y comprometerse con este repositorio.

Inicialice este repositorio con:

☐ Agregar un archivo README
Aquí puedes escribir una descripción detallada de tu proyecto. [Obtén más información sobre los archivos README.](#)

Agregar .gitignore


Plantilla .gitignore: Ninguna

Seleccione los archivos que no desea rastrear de una lista de plantillas. [Obtenga más información sobre cómo ignorar archivos.](#)

Elija una licencia

Licencia : Ninguna

Una licencia indica a otros qué pueden y no pueden hacer con tu código. [Obtén más información sobre las licencias.](#)

 Estás creando un repositorio público en tu cuenta personal.

[Crear repositorio](#)

Luego de completar todos los campos hay que oprimir el botón Create Repository /

Crear Repositorio.

Una vez creado el repositorio nos va a mostrar una url la cual vamos a usar en nuestro repositorio local y vamos a configurar para que se comuniquen con nuestro repositorio remoto.

El siguiente paso es abrir Git e inicializarlo con el comando `git init`, esto va a crear un repositorio vacío de Git en la dirección de nuestro proyecto, antes de conectar con nuestro repositorio remoto tenemos que identificarnos mediante el siguiente comando `git config user.name "nombre"`, luego hay que configurar el email asociado con el nombre de usuario con el comando `git config user.email "usuario@gmail.com"`.

Una vez hecho eso comunicamos nuestro repositorio local con el remoto creado en GitHub mediante el siguiente comando `git remote add origin "..."` entre las comillas va la url de nuestro repositorio de GitHub, por último le damos enter.

3. ¿Cómo crear una rama en Git?

Para crear una nueva rama en Git usamos el comando `git branch` y colocamos el nombre de rama que queremos crear, ejemplo `git branch newBranch`.

Para saber en qué rama estamos usamos el comando `git branch`.

4. ¿Cómo cambiar a una rama en Git?

Para cambiar a una rama en Git podemos usar la forma más clásica usando el comando `git checkout new branch`. Podemos corroborarlo con `git branch`. Además se puede utilizar otra forma de moverse entre ramas, la cual es muy recomendada, se puede hacer mediante el comando `git switch newBranch`.

5. ¿Cómo fusionar ramas en Git?

Para fusionar ramas en Git lo primero que hay que hacer es posicionarnos en la rama a la cual queremos fusionar los cambios con `git checkout master`, luego utilizamos el comando `git merge newBranch` para fusionar la rama principal con la actual. Esto va a hacer que todo lo que tengamos generado en la rama newBranch se vuelva a la rama principal.

6. ¿Cómo crear un commit en Git?

Un commit en Git es la forma que tenemos de registrar los cambios que realicemos, para luego tener puntos a los cuales podamos regresar o ver su cambio en cada momento. Para hacer un commit primero hay que realizar algún cambio el cual queramos guardar, luego necesitamos usar el comando `git add` para incluir los cambios efectuados del o de los archivos en el commit, ejemplo `git add "carpeta1"` (para agregar un archivo específico) o `git add` (para agregar todos los archivos).

Una vez hecho lo anterior podemos hacer el commit con el siguiente comando `git commit -m "Mensaje de los cambios"` (entre comillas ponemos un mensaje breve del cambio que hicimos).

7. ¿Cómo enviar un commit a GitHub?

Para enviar un commit a GitHub tenemos que empujarlo con el comando `git push -u origin nombreRama`, pero antes debemos realizar cambios en nuestro repositorio y

utilizar los comandos `git add` . y `git commit -m "Mensaje de cambios hechos"` para guardar los cambios.

8. ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión de nuestro repositorio, el cual se encuentra subido en un servidor, además de que pueden acceder a este varios usuarios mediante internet, siempre y cuando se encuentre en modo público. Los repositorios remotos nos permiten trabajar en diferentes proyectos de forma colaborativa, compartir cambios y tener una copia de seguridad del código.

9. ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git debemos utilizar el comando `git remote add`, este va a vincular el repositorio remoto con el local, además se le debe asignar un nombre para identificarlo más fácilmente, ejemplo `git remote add pruebaGit https://github.com/usuario/pruebaGit.git` , de esta forma ya podemos comenzar a trabajar con él. Además si queremos ver la lista de repositorios remotos podemos usar el comando `git remote -v` .

10. ¿Cómo empujar cambios a un repositorio remoto?

Primero hay que guardar los cambios hechos con `git commit` y debemos tener vinculado el repositorio remoto de GitHub con el repositorio local, esto lo hacemos con `git remote add pruebaGit`, una vez que hayamos realizado lo anterior podemos empujar los cambios al repositorio remoto con el comando `git push origin pruebaGit` .

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

Para tirar los cambios de un repositorio remoto podemos usar el comando `git pull origin pruebaGit`, hecho esto podremos descargar y fusionar los cambios del repositorio remoto de GitHub en nuestra rama local.

12. ¿Qué es un fork de repositorio?

Un fork o bifurcación es una copia que hace GitHub a un repositorio remoto de nuestro interés en nuestra cuenta de forma totalmente independiente y todos los cambios que hagamos no van a afectar al repositorio original, sino que van a estar asociados a nuestra cuenta. Esta es una herramienta que se utiliza para colaborar en proyectos de código abierto

13. ¿Cómo crear un fork de un repositorio?

Para crear un fork de un repositorio primero hay que buscar el botón **Fork** y darle clic, esto va a crear una copia en nuestra cuenta de GitHub, la cual va a ser independiente del repositorio original, los cambios que hagamos se verán solo en nuestra copia, por último debemos clonar el repositorio, guardarlo en una carpeta y luego bajarlo a nuestro repositorio local, así podemos hacer modificaciones que consideremos que sean de ayuda o mejora para código del repositorio.

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

Para enviar una solicitud de extracción o Pull Request a un repositorio, lo primero que hay que hacer es clicar en la parte superior donde dice **Pull request**, luego nos va a aparecer la opción **new pull request** la cual debemos dar clic y allí veremos los cambios que hemos hecho comparándolos con el repositorio original, hay que escribir una descripción clara de los cambios que realizamos y cómo serían de ayuda o mejora para el código original, por último clickeamos **Create pull request**.

15. ¿Cómo aceptar una solicitud de extracción?

Una vez realizada la solicitud de extracción, el autor del repositorio revisará los cambios que hemos realizado y si nuestros cambios no interfieren con el código original y son de ayuda, entonces el autor podrá aceptarla clickeando en **Merge pull request**, es así que sumará a su repositorio los cambios que hemos sugerido.

16. ¿Qué es una etiqueta en Git?

En Git una etiqueta es una referencia que se usa para marcar puntos específicos del historial de un repositorio, es una forma conveniente de acceder de forma rápida a un commit específico, se utilizan generalmente para marcar versiones de un proyecto, además de que una vez creada no puede cambiarse.

17. ¿Cómo crear una etiqueta en Git?

Hay dos formas de crear una etiqueta en Git, están las etiquetas ligeras y las etiquetas anotadas.

Para crear una etiqueta ligera se usa el comando `git tag nombreEtiqueta`, esta no lleva información adicional como el nombre del autor o la fecha.

Para crear una etiqueta anotada se usa el comando `git tag -a nombreEtiqueta -m "Mensaje de la etiqueta"`, esta es más completa, ya que almacena información adicional como el nombre del autor, fecha y un mensaje asociado.

18. ¿Cómo enviar una etiqueta a GitHub?

Lo primero que debemos hacer es crear una etiqueta, que puede ser ligera o anotada, esta debemos crearla en nuestro repositorio local, una vez hecho esto tenemos que empujarla al repositorio remoto de GitHub con el comando `git push origin nombreEtiqueta`, si lo que queremos es empujar todas las etiquetas que hemos creado entonces podemos usar el comando `git push origin --tags`.

19. ¿Qué es un historial de Git?

Un historial de Git es una secuencia de todos los cambios (commits) que hemos hecho en nuestro repositorio a lo largo del tiempo. En el historial de Git cada commit realizado contiene información sobre los cambios hechos, el autor de esos cambios, su fecha y un mensaje donde se describe lo que fué modificado. Al ser tan detallado se puede realizar un seguimiento de la evolución de un proyecto, permitiendo que las tareas como depuración, colaboración o reversión a versiones anteriores del código sean más sencillas.

20. ¿Cómo ver el historial de Git?

Para ver el historial de Git se puede usar el comando `git log`, este comando nos mostrará los commits en orden cronológico inverso, del más reciente al más antiguo. Cada commit nos mostrará el hash del commit (identificador único), el autor del commit (nombre y correo electrónico), la fecha del commit y el mensaje del commit. Se puede ver de forma completa con el comando `git log --oneline` o de forma gráfica con el comando `git log --oneline --graph`.

21. ¿Cómo buscar en el historial de Git?

Para buscar en el historial de Git, podemos usar varios comandos que nos van a permitir filtrar y localizar commits específicos.

Para buscar commits que tengan una palabra o frase específica en el mensaje de commit, se puede utilizar `git log` con la opción `--grep: git log --grep="palabra clave"`

Para buscar commits que han modificado un archivo específico, usa `git log` seguido

del nombre del archivo:

```
git log -- nombre_del_archivo
```

Para buscar commits en un rango de fechas específico, usamos las opciones `--since` y `--until`:

```
git log --since="2025-01-01" --until="2025-01-31"
```

Para encontrar commits hechos por un autor específico, se puede usar `--author`:

```
git log --author="Nombre del Autor"
```

22. ¿Cómo borrar el historial de Git?

Para borrar el historial de Git utilizamos el comando `git reset`, debemos aclarar que existen distintas formas de utilizarlo, una vez que entendamos cada una de ellas podremos comenzar a utilizarlas de forma correcta sin correr peligro de borrar algo por error.

Formas de utilizar el git reset:

- `git reset` (elimina todos los archivos y carpetas del proyecto)
- `git reset nombre-archivo` (quita el archivo indicado)
- `git reset nombre-carpeta` (quita todos los archivos de esa carpeta)
- `git reset nombre-carpeta/nombre-archivo` (quita ese archivo que a la vez está dentro de la carpeta especificada)
- `git reset nombre-carpeta/*.extensión` (quita todos los archivos que cumplan con la condición indicada)


23. ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub no es más que un tipo de repositorio donde sólo algunos usuarios autorizados tienen acceso, es muy útil para proyectos que aún están en desarrollo y no queremos que otros usuarios lo modifiquen.

24. ¿Cómo crear un repositorio privado en GitHub?

Para crear un repositorio privado en GitHub se debe oprimir el botón **New**, al hacerlo nos va a llevar a crear el nuevo repositorio donde debemos llenar una serie de campos que tienen que ver con el nombre, descripción de lo que se va a subir, si queremos que sea público o privado, etc.


Dueño * Nombre del repositorio *


 Joygamarra09 /

Los buenos nombres de repositorios son cortos y fáciles de recordar. ¿Necesitas inspiración? ¿Qué te parece?

[enigma especial ?](#)

Descripción (opcional)


☐  Público
Cualquier persona en internet puede ver este repositorio. Tú decides quién puede contribuir.

☒  Privado
Tú eliges quién puede ver y comprometerse con este repositorio.

Inicialice este repositorio con:


☐ Agregar un archivo README
Aquí puedes escribir una descripción detallada de tu proyecto. [Obtén más información sobre los archivos README.](#)

Agregar .gitignore


Plantilla .gitignore: Ninguna 

Seleccione los archivos que no desea rastrear de una lista de plantillas. [Obtenga más información sobre cómo ignorar archivos.](#)

Elija una licencia

Licencia: Ninguna 

Una licencia indica a otros qué pueden y no pueden hacer con tu código. [Obtén más información sobre las licencias.](#)

 Estás creando un repositorio privado en tu cuenta personal.

[Crear repositorio](#)

Hay que clicar donde dice Privado y luego se oprime el botón **Create Repository / Crear Repositorio**.

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

Para invitar a alguien a un repositorio privado en GitHub hay que:

- ingresar al repositorio y hacer clic en la pestaña que dice **Settings**
- Oprimir la opción **Collaborators** en el menú que se encuentra del lado izquierdo
- Una vez hecho esto se nos abrirá la página donde se puede administrar los colaboradores y allí hay que hacer clic en el botón **Add people** e ingresa el nombre de usuario de GitHub de la persona que deseas invitar. Selecciona el nivel de acceso que quieres otorgar: **Read, Triage, Write, Maintain**, o **Admin**. Haz clic en el botón **Add** para enviar la invitación.


26. ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un tipo de repositorio en el cual se puede acceder desde internet y no necesita de ningún permiso, ya que es accesible para cualquier persona y ésta ver, clonar y contribuir al proyecto si el autor lo permite.

27. ¿Cómo crear un repositorio público en GitHub?

Para crear un repositorio público en GitHub se debe oprimir el botón **New**, al hacerlo nos va a llevar a crear el nuevo repositorio donde debemos llenar una serie de campos que tienen que ver con el nombre, descripción de lo que se va a subir, si queremos que sea público o privado, etc.


Dueño * **Nombre del repositorio ***


 Joygamarra09 /

Los buenos nombres de repositorios son cortos y fáciles de recordar. ¿Necesitas inspiración? ¿Qué te parece?

[carnaval de risas ?](#)

Descripción (opcional)

☒  **Público**
Cualquier persona en internet puede ver este repositorio. Tú decides quién puede contribuir.

☐  **Privado**
Tú eliges quién puede ver y comprometerse con este repositorio.

Inicialice este repositorio con:

☐ Agregar un archivo README
Aquí puedes escribir una descripción detallada de tu proyecto. [Obtén más información sobre los archivos README.](#)

Agregar .gitignore


Plantilla .gitignore: Ninguna ▾

Seleccione los archivos que no desea rastrear de una lista de plantillas. [Obtenga más información sobre cómo ignorar archivos.](#)

Elija una licencia

Licencia : Ninguna ▾

Una licencia indica a otros qué pueden y no pueden hacer con tu código. [Obtén más información sobre las licencias.](#)

 Estás creando un repositorio público en tu cuenta personal.

[Crear repositorio](#)

Hay que clicar donde dice Público y luego se oprime el botón Create Repository / Crear Repositorio.

28. ¿Cómo compartir un repositorio público en GitHub?

Para compartir un repositorio público en GitHub hay que ingresar al repositorio que queremos compartir, luego oprimimos el botón que se encuentra en la parte superior izquierda que dice **<> Code** y copiamos la URL que nos aparece, por último compartimos la URL con quien queramos compartir el repositorio.

2) Realizar la siguiente actividad:


- Crear un repositorio.
 - Dale un nombre al repositorio.
 - Elige que el repositorio sea público.
 - Inicializa el repositorio con un archivo.

Crear un nuevo repositorio

Un repositorio contiene todos los archivos del proyecto, incluido el historial de revisiones. ¿Ya tienes un repositorio de proyectos en otro lugar? [Importa un repositorio](#).

Los campos obligatorios están marcados con un asterisco (*).

Dueño *

 Joygamarra09

Nombre del repositorio *

ProgramacionU2

ProgramacionU2 está disponible.

Los buenos nombres de repositorios son cortos y fáciles de recordar. ¿Necesitas inspiración? ¿Qué te parece? **periquito mejorado** ?

Descripción (opcional)

Mi primer repositorio de Programación 1.



Público

Cualquier persona en internet puede ver este repositorio. Tú decides quién puede contribuir.



Privado

Tú eliges quién puede ver y comprometerse con este repositorio.

Inicialice este repositorio con:



Agregar un archivo README

Aquí puedes escribir una descripción detallada de tu proyecto. [Obtén más información sobre los archivos README](#).

Agregar .gitignore

Plantilla .gitignore: Ninguna

Seleccione los archivos que no desea rastrear de una lista de plantillas. [Obtenga más información sobre cómo ignorar archivos](#).

Elija una licencia

Licencia: Ninguna

Una licencia indica a otros qué pueden y no pueden hacer con tu código. [Obtén más información sobre las licencias](#).

Esto establecerá **principal** Como rama predeterminada. Cambia el nombre predeterminado en la [configuración](#).



Estás creando un repositorio público en tu cuenta personal.

Crear repositorio

• 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.
- Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).

Realizamos los comandos `git add .` y `git commit`, luego subimos los cambios con `git push origin master`.


```
joana@DESKTOP-1G5FAU4 MINGW64 ~/Desktop/TPs Programación (master)
$ git init
Reinitialized existing Git repository in C:/Users/joana/Desktop/TPs Programación/.git/

joana@DESKTOP-1G5FAU4 MINGW64 ~/Desktop/TPs Programación (master)
$ git add .

joana@DESKTOP-1G5FAU4 MINGW64 ~/Desktop/TPs Programación (master)
$ git commit -m "Agregando cambios a mi ejercicio2"
[master 18ca39c] Agregando cambios a mi ejercicio2
1 file changed, 1 insertion(+), 5 deletions(-)

joana@DESKTOP-1G5FAU4 MINGW64 ~/Desktop/TPs Programación (master)
$ git push ejercicio2 master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 850 bytes | 425.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
remote:

remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/Joygamarra09/ProgramacionU2/pull/new/master
remote:
To https://github.com/Joygamarra09/ProgramacionU2.git
 * [new branch]      master -> master

joana@DESKTOP-1G5FAU4 MINGW64 ~/Desktop/TPs Programación (master)
$

joana@DESKTOP-1G5FAU4 MINGW64 ~/Desktop/TPs Programación (master)
$ git push -u ejercicio2 master
branch 'master' set up to track 'ejercicio2/master'.
Everything up-to-date
```

Verificamos que se haya subido correctamente a GitHub

principal
2 sucursales
0 etiquetas
Ir al archivo
Agregar archivo
Código

Joygamarra09
Agregando ejercicio2.py
5427 dBf · ahora
4 confirmaciones

README.md	Compromiso inicial	hace 1 hora
ejercicio2.py	ejercicio2.py	hace 1 minuto

• Creando Branchs

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

Creamos una branch

```
PS C:\Users\joana\Desktop\TPs Programación> git branch nuevaRama
PS C:\Users\joana\Desktop\TPs Programación> git branch
* master
  nuevaRama
PS C:\Users\joana\Desktop\TPs Programación>
```

```
PS C:\Users\joana\Desktop\TPs Programación> git checkout nuevaRama
Switched to branch 'nuevaRama'
PS C:\Users\joana\Desktop\TPs Programación> git branch
master
* nuevaRama
PS C:\Users\joana\Desktop\TPs Programación> █
```

Creamos un archivo nuevo

```
PS C:\Users\joana\Desktop\TPs Programación> git add .
PS C:\Users\joana\Desktop\TPs Programación> git commit -m "Se creó ejercicio3"
[nuevaRama bd8c0cb] Se creó ejercicio3
1 file changed, 3 insertions(+)
create mode 100644 ejercicio3.py
```

TPS PROGRAMACIÓN ejercicio2.py ejercicio3.py	ejercicio3.py > ... <pre>1 2 nombre = input("Ingrese su nombre completo: ") 3 print(f"Te damos la bienvenida {nombre}.")</pre>
-----------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------

Subimos la branch

```
PS C:\Users\joana\Desktop\TPs Programación> git push
origin nuevaRama
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 627 bytes | 313.00 KiB/s, done.
Total 6 (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/Joygamarra09/ProgramacionU2/pull/new/nuevaRama
remote:
To https://github.com/Joygamarra09/ProgramacionU2.git * [new branch]      nuevaRama -> nuevaRama
PS C:\Users\joana\Desktop\TPs Programación> █
```

main ▾ 3 Branches 0 Tags		<input type="text" value="Go to file"/> <input type="button" value="Add file"/> <input type="button" value="Code"/>
<div> Joygamarra09 ejercicio3.py b29be3a · now 5 Commits </div>		
README.md	Initial commit	2 hours ago
ejercicio2.py	ejercicio2.py	50 minutes ago
ejercicio3.py	ejercicio3.py	now

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".

main 1 Branch 0 Tags

Go to file

Add file

Code

Joygamarra09 Initial commit

c0471d8 · now 1 Commit

README.md

Initial commit

now

README

conflict-exercise

Este repositorio pertenece al ejercicio 3 del TP1 de la Unidad 2

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
```

```
PS C:\Users\joana\Desktop\U2 TP ejercicio3> git clone https://github.com/Joygamarra09/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.
PS C:\Users\joana\Desktop\U2 TP ejercicio3>
```

- Entra en el directorio del repositorio:

```
cd conflict-exercise
```

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:

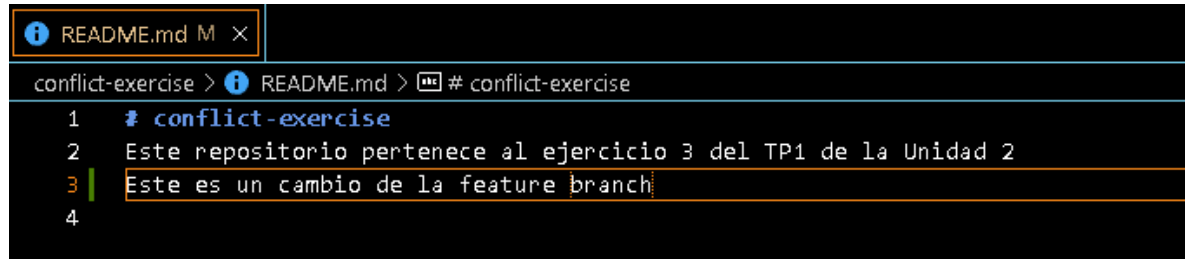
```
git checkout -b feature-branch
```

```
PS C:\Users\joana\Desktop\U2 TP ejercicio3> cd conflict-exercise
PS C:\Users\joana\Desktop\U2 TP ejercicio3\conflict-exercise> git branch feature-br
ch
PS C:\Users\joana\Desktop\U2 TP ejercicio3\conflict-exercise> git branch
feature-branch
* main
PS C:\Users\joana\Desktop\U2 TP ejercicio3\conflict-exercise>
```

```
PS C:\Users\joana\Desktop\U2 TP ejercicio3> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\joana\Desktop\U2 TP ejercicio3> 
```

- 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.



```
conflict-exercise > README.md > # conflict-exercise
1  # conflict-exercise
2  Este repositorio pertenece al ejercicio 3 del TP1 de la Unidad 2
3  Este es un cambio de la feature branch
4 
```

- Guarda los cambios y haz un commit:

```
git add README.md
```

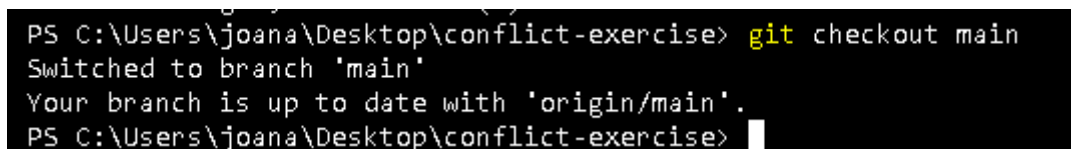
```
git commit -m "Added a line in feature-branch"
```

```
PS C:\Users\joana\Desktop\conflict-exercise> git checkout feature-branch
Switched to branch 'feature-branch'
PS C:\Users\joana\Desktop\conflict-exercise> git add README.md
PS C:\Users\joana\Desktop\conflict-exercise> git commit -m "Added a line in feat
ure-branch"
[feature-branch 5f7c582] Added a line in feature-branch
1 file changed, 1 insertion(+)
```

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

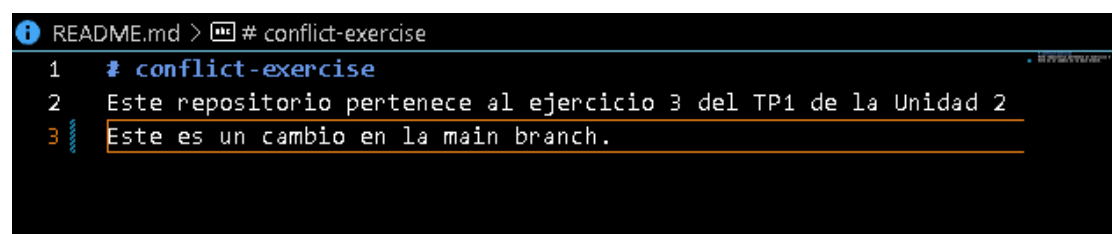
- Cambia de vuelta a la rama principal (main):

```
git checkout main
```



```
PS C:\Users\joana\Desktop\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\joana\Desktop\conflict-exercise> 
```

- Edita el archivo README.md de nuevo, añadiendo una línea diferente: Este es un cambio en la main branch.



```
README.md > # conflict-exercise
1  # conflict-exercise
2  Este repositorio pertenece al ejercicio 3 del TP1 de la Unidad 2
3  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"
```

```
PS C:\Users\joana\Desktop\conflict-exercise> git add README.md
PS C:\Users\joana\Desktop\conflict-exercise> git commit -m "Added a line in main branch"
[main efd2d93] Added a line in main branch
1 file changed, 1 insertion(+)
PS C:\Users\joana\Desktop\conflict-exercise>
```

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.

```
PS C:\Users\joana\Desktop\conflict-exercise> git checkout main
Already on 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)
PS C:\Users\joana\Desktop\conflict-exercise> 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.
PS C:\Users\joana\Desktop\conflict-exercise>
```

Paso 6: Resolver el conflicto

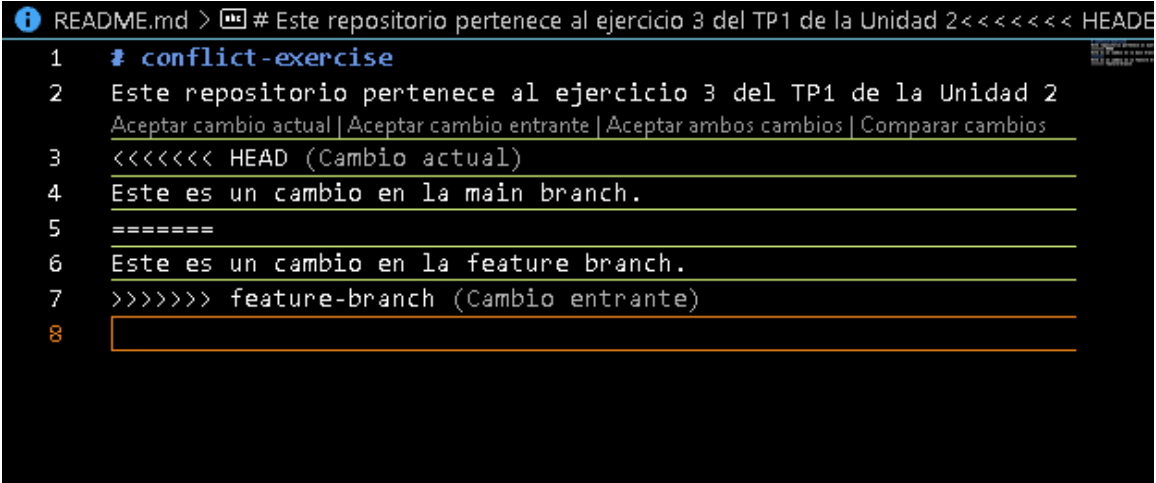
- Abre el archivo README.md en tu editor de texto. Verás algo similar a esto: `<<<<<< HEAD`

Este es un cambio en la main branch.

```
=====
```

Este es un cambio en la feature branch.

```
>>>>>> feature-branch
```



```
1 # conflict-exercise
2 Este repositorio pertenece al ejercicio 3 del TP1 de la Unidad 2
3 <<<<<< HEAD (Cambio actual)
4 Este es un cambio en la main branch.
5 =====
6 Este es un cambio en la feature branch.
7 >>>>>> feature-branch (Cambio entrante)
8
```

- 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).

```
README.md > # conflict-exercise
1 # conflict-exercise
2 Este repositorio pertenece al ejercicio 3 del TP1 de la Unidad 2
3 Este es un cambio en la main branch.
4 Este es un cambio en la feature branch.
5
```

- Añade el archivo resuelto y completa el merge:

`git add README.md`

`git commit -m "Resolved merge conflict"`

```
PS C:\Users\joana\Desktop\conflict-exercise> git add README.md
PS C:\Users\joana\Desktop\conflict-exercise> git commit -m "Resolved merge conflict"
[main 7dbe125] Resolved merge conflict
PS C:\Users\joana\Desktop\conflict-exercise>
```

Paso 7: Subir los cambios a GitHub

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

GitHub: `git push origin main`

```
PS C:\Users\joana\Desktop\conflict-exercise> 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), 816 bytes | 272.00 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/Joygamarra09/conflict-exercise.git
   c0471d8..7dbe125  main -> main
PS C:\Users\joana\Desktop\conflict-exercise>
```

- También sube la feature-branch si deseas:

`git push origin feature-branch`

```
PS C:\Users\joana\Desktop\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/Joygamarra09/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/Joygamarra09/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch
PS C:\Users\joana\Desktop\conflict-exercise>
```

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.



1 archivo
cambiado

+ -
2 1 líneas cambiadas

Buscar dentro del código



▼ README.md

+2 -1



@@ -1,3 +1,4 @@

```
1 # conflict-exercise
2 Este repositorio pertenece al
  ejercicio 3 del TP1 de la Unidad 2
3 - Este es un cambio en la main
  branch.
```

```
1 # conflict-exercise
2 Este repositorio pertenece al
  ejercicio 3 del TP1 de la Unidad 2
3 + Este es un cambio en la main
  branch.
4 + Este es un cambio en la feature
  branch.
```