

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

Programación I

Práctico 2: Git y GitHubActividades

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

- ¿Qué es GitHub?

GitHub es una plataforma donde los programadores pueden guardar, organizar y compartir su código de manera segura. Funciona con Git, un sistema que permite hacer seguimiento de los cambios en los archivos, facilitando la colaboración entre varias personas en un mismo proyecto. Es gratuita y de código abierto.

- ¿Cómo crear un repositorio en GitHub?

Una vez creada tu cuenta en GitHub, puedes empezar a crear un nuevo repositorio:

Como va a ser nuestro primer repositorio, debemos apretar el botón con la descripción "CREATE REPOSITORY".

Completamos colocando un nombre al repositorio.

Elegimos si queremos que el repositorio sea PÚBLICO o PRIVADO.

Dejamos una descripción del mismo (es opcional).

Generalmente se tilda la opción "Add Readme File", en la cual dejamos información acerca del proyecto, podría ser un resumen del mismo.

Por último apretamos "create repository" y terminamos de generarlo.

- ¿Cómo crear una rama en Git?

Para crear una rama en git, nos posicionamos en la rama principal (main o master), y luego en consola colocamos el siguiente comando: `git checkout` "el nombre de la rama nueva", le damos "enter" y se genera la nueva rama (branch).

Ej: `git checkout ejercicio1`

- ¿Cómo cambiar a una rama en Git?

Para cambiar de rama, en consola ponemos `git branch` y nos va a mostrar todas las ramas (incluyendo la principal), en la cual aparece con un * del lado izquierdo, la rama que está seleccionada en ese momento. Ahora bien, queremos cambiar de

rama, tenemos que poner: git checkout + nombre de la raíz que queremos seleccionar.

- **¿Cómo fusionar ramas en Git?**

Para fusionar ramas, seleccionamos la rama que queremos fusionar, y una vez seleccionada, ponemos: git merge “nombre de la otra rama que queremos fusionar”.

- **¿Cómo crear un commit en Git?**

El “commit” es como una foto de lo que hemos realizado hasta ese preciso en el código, cuando realizamos el commit, estaríamos guardando los cambios que hemos realizado y les agrega un comentario.

Como se realiza esto?, en consola colocamos: git commit -m “el mensaje que le queremos poner”

- **¿Cómo enviar un commit a GitHub?**

Podremos hacer esto de la siguiente manera:

Inicializamos git → (git init)

#Seleccionamos el archivo que queremos enviar

Ponemos → (git add .) y con esto estamos seleccionando todos los archivos de la carpeta actual para realizar el commit.

En caso de querer incluir un archivo específico, solo basta con poner → (git add nombre-del-archivo.txt)

Creamos el commit: → (git commit -m "Mensaje descriptivo del cambio")

Nos conectamos a GITHUB:

En caso de no haberlo hecho antes, para conectarnos al repositorio remoto, debemos pegar el siguiente código: (git remote add origin

<https://github.com/tu-usuario/tu-repo.git>), que nos haría una clonación de GitHub a nuestra máquina local.

Enviamos el commit a GITHUB:

Colocamos lo siguiente → (git push origin main), en caso de que nuestra rama principal se llame master, reemplazar main por master.

Aclaracion: Si es la primera vez que realizas el “push”, puede que tengas que configurar tus credenciales de la siguiente manera:

- git config --global user.name "nombre"
- git config --global user.email "tu email "

- **¿Qué es un repositorio remoto?**

Un repositorio remoto es un espacio de almacenamiento que está ubicado en el servidor o en la nube donde generalmente es usado para que podamos subir nuestros proyectos y que podamos colaborar con otros desarrolladores.

Existen plataformas como GitHub, GitLab, que ofrecen servicios para gestionar los repositorios remotos.

Para que sirve?, para que podamos sincronizar los cambios realizados en un repositorio local con el remoto, compartir trabajo, ir a otro sitio y en otra PC, poder seguir trabajando sin la necesidad de llevar tu PC o la del trabajo.

- **¿Cómo agregar un repositorio remoto a Git?**

Lo que se hace para agregar un repositorio remoto a git, es colocar: `git remote add` (nombre del remoto y la url).

Verificamos que se agregó correctamente el repositorio remoto: (`git remote -v`)

Que hace ese código?, nos dará una lista de todos los repositorios remotos que tenemos vinculados, por lo que si aparece allí, estaría creado correctamente.

- **¿Cómo empujar cambios a un repositorio remoto?**

Para “empujar” nuestro código del repositorio local, al repositorio remoto, hacemos lo siguiente, colocamos: `git push -u origin “nombre de la rama”`.

Donde origin es el nombre predeterminado del repositorio remoto y `-u` se utiliza para establecer una relación entre la rama local y la rama remota, generalmente se coloca una vez.

- **¿Cómo tirar de cambios de un repositorio remoto?**

Primero lo que hacemos es ejecutar el comando : (`git pull origin “nombre de la rama”`)

Como ejemplo si queremos traer los cambios de una rama llamada ejercicio1, pondríamos lo siguiente:

`git pull origin ejercicio1`

- **¿Qué es un fork de repositorio?**

Un fork seria como “copiar y pegar” un proyecto, por ejemplo quiero copiar el proyecto de un usuario, podría copiar ese proyecto utilizando el fork, para poder hacer cambios en mi computadora, modificando ese proyecto en mi PC aunque no modificaria el proyecto original del usuario.

Es en si una duplicación de un proyecto.

- ¿Cómo crear un fork de un repositorio?

Para crear un fork de un repositorio, primero voy al repositorio en GitHub, y accedo al repositorio que quiero Forkar. Luego apreto el botón FORK, en la pagina superior derecha. Al hacer eso, se va a crear una copia de ese repositorio en mi cuenta de GitHub.

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

Primero se hace un fork del repositorio original, se clona el FORK a la PC, como se hace esto?, en la terminal se coloca lo siguiente → (git clone https://github.com/usuario/nombre_de_repositorio.git)

Luego entramos en la carpeta: cd **nombre_de_repositorio**

Creamos una nueva rama para los cambios → (git checkout -b “nombre de la nueva rama”

Realizamos los cambios que deseamos y escribimos → git add .

Realizamos un commit con el mensaje que quremos de los nuevos cambios → git commit -m “cambios realizados...”

Subimos la rama a GitHub → git push origin **nombre_de_repositorio**

Por ultimo creamos el Pull Request en Github:

Ir al repositorio en GitHub

Hacer click en “Compare and pull request”

Podemos escribir una descripcion

Y clickeamos en “create pull request”

- ¿Cómo aceptar una solicitud de extracción (Pull Request)?

Vamos al repositorio en GitHub donde nos hicieron la solicitud de extracción.

Hacer click en pull request que queramos revisar, dentro de la pestaña “Pull Request” en la cual se muestran todas las solicitudes.

Una vez que le damos click, podremos ver los cambios que nos proponen, comentarios del autor de la solicitud y los archivos que fueron modificados.

Obviamente tenemos que chequear los cambios, los archivos que fueron modificados, etc.

Para probar dichos cambios, debemos realizar un checkout de la rama del pull request:

git checkout **nombre_de_repositorio** corroborando que funciona correctamente.

Luego de que estemos de acuerdo con dichos cambios, aceptamos la solicitud de extracción, se hace de la siguiente manera:

Click en “Merge pull request”

Escribimos un mensaje si lo deseamos

Click en “Confirm merge”

Como buena practica se borra la rama del pull request si el merge fue exitoso, para mantener nuestro repositorio “limpio”.

- ¿Que es un etiqueta en Git?

Es una referencia o etiqueta inmóvil, a un punto que especificamos en el historial de commits. Generalmente utilizadas para identificar versiones importantes de un proyecto.

- ¿Cómo crear una etiqueta en Git?

Basta con colocar → git tag “nombre de la etiqueta”

Ej: git tag V2.3

- ¿Cómo enviar una etiqueta a GitHub?

Podríamos usar → git tag (para ver las etiquetas que tenemos)

Luego se utiliza → git push “nombre de la etiqueta que queremos enviar”

Y → git push - -tags para enviar varias etiquetas y subirlas a todas de una sola vez.

- ¿Qué es un historial de Git?

Es el registro completo de los cambios realizados en un repositorio

- ¿Cómo ver el historial de Git?

Solo basta con poner el comando: git log

- ¿Cómo buscar en el historial de Git?

Hay varias maneras de buscar en el historial, se puede buscar por:

Autor → git log --autor = “nombre”

Palabra clave → git log --grep=”palabra clave”

Cambios en un archivo específico → git log -- "nombre del archivo”

- ¿Cómo borrar el historial de Git?

Solo basta con borrar la carpeta .git (que está oculta en la carpeta de nuestro proyecto) o también poniendo el comando rm -rf .git

- ¿Qué es un repositorio privado en GitHub?

Repositorio al que solo tú y las personas que autorices pueden acceder. No es visible públicamente.

- ¿Cómo crear un repositorio privado en GitHub?

Una vez en GitHub, ir a la parte superior derecha y haz clic en el ícono "+" y selecciona "New repository".

Escribir el nombre del repositorio, puedes agregar la descripción, selecciona "Private" y listo, generaste un repositorio privado, donde solo tu y los colaboradores que desees podrán verlo.

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

Una vez dentro de GitHub, abrimos el repositorio que queremos, hacemos click en la pestaña "Settings" → seleccionamos "Collaborators and teams" o "Manage acces" → click en "Add people" → escribimos el nombre o correo de la persona que queremos invitar → seleccionamos su perfil → click en "Add"

Aclaración: La persona debe aceptar la invitación dándole "Accept invitation" para unirse.

- ¿Qué es un repositorio público en GitHub?

El repositorio público es un repositorio al que cualquier persona puede acceder, o sea todo lo que contiene dicho repositorio será visible para el público, como por ejemplo sus archivos, el historial de commits, el código, etc

El nivel de acceso depende de como esté configurado el repositorio, puede estar configurado en Lectura (ver y clonar) o Contribución (solo los colaboradores autorizados pueden hacer cambios directamente), otros solo pueden proponer cambios enviando un pull request.

- ¿Cómo crear un repositorio público en GitHub?

En GitHub, seleccionamos "+" "New repository"
Configuramos nombre, descripción (opcional), y seleccionamos "Public"

- ¿Cómo compartir un repositorio público en GitHub?

Dentro de GitHub, vamos a nuestro repositorio → click en "code"
→ Copiamos el enlace del repositorio

➔ Se puede enviar dicho enlace por redes sociales, correo o incluirlo en nuestro website.

2) Realizar la siguiente actividad:

Crear un repositorio.

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

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).

Creando Branchs

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

https://github.com/PaSe1982/ejercicio2_repositorio.git


The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there's a link to 'Import a repository'. A note states 'Required fields are marked with an asterisk (*)'. The form has two main sections: 'Owner' and 'Repository name'. The 'Owner' dropdown is set to 'PaSe1982'. The 'Repository name' input field contains 'ejercicio2_repositorio', and a green checkmark indicates it is available. Below this, there's a suggestion for repository names: 'Great repository names are short and memorable. Need inspiration? How about verbose-invention?'. The 'Description' field is optional and contains the text 'Trabajo Practico 2 - Ejercicio 2 - VENTURA PABLO'. At the bottom, there are two radio buttons for visibility: 'Public' (selected) and 'Private'. The 'Public' option is described as 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option is described as 'You choose who can see and commit to this 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 ***

 **PaSe1982** / **ejercicio2_repositorio**

✓ **ejercicio2_repositorio** is available.

Great repository names are short and memorable. Need inspiration? How about **verbose-invention** ?

Description (optional)

Trabajo Practico 2 - Ejercicio 2 - VENTURA PABLO

☒ **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.

```

MINGW64:/c:/Users/Pablo/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (main)
$ git add .

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (main)
$ git commit -m "Agregando archivos"
[main e33c95f] Agregando archivos
 1 file changed, 1 insertion(+)
 create mode 100644 nuevo_archivo.txt

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (main)
$ git push origin main
info: please complete authentication in your browser...
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 326 bytes | 326.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/PaSe1982/ejercicio2_repositorio.git
   14e354b..e33c95f  main -> main

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (main)
$ |

```

```

MINGW64:/c:/Users/Pablo/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (main)
$ git branch ramal

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (main)
$ git branch
* main
  ramal

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (main)
$ git checkout ramal
Switched to branch 'ramal'

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (ramal)
$ git add .

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (ramal)
$ git commit -m "Se hizo modificacion al txt"
On branch ramal
nothing to commit, working tree clean

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (ramal)
$ git status
On branch ramal
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   nuevo_archivo.txt

no changes added to commit (use "git add" and/or "git commit -a")

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (ramal)
$ git add .

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (ramal)
$ git commit -m "Se hizo modificacion al txt"
[ramal 2298970] Se hizo modificacion al txt
 1 file changed, 2 insertions(+), 1 deletion(-)

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (ramal)
$ git push -u origin ramal
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 364 bytes | 364.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Create a pull request for 'ramal' on GitHub by visiting:
remote:   https://github.com/PaSe1982/ejercicio2_repositorio/pull/new/ramal
remote:
To https://github.com/PaSe1982/ejercicio2_repositorio.git
 * [new branch] ramal -> ramal
branch 'ramal' set up to track 'origin/ramal'.

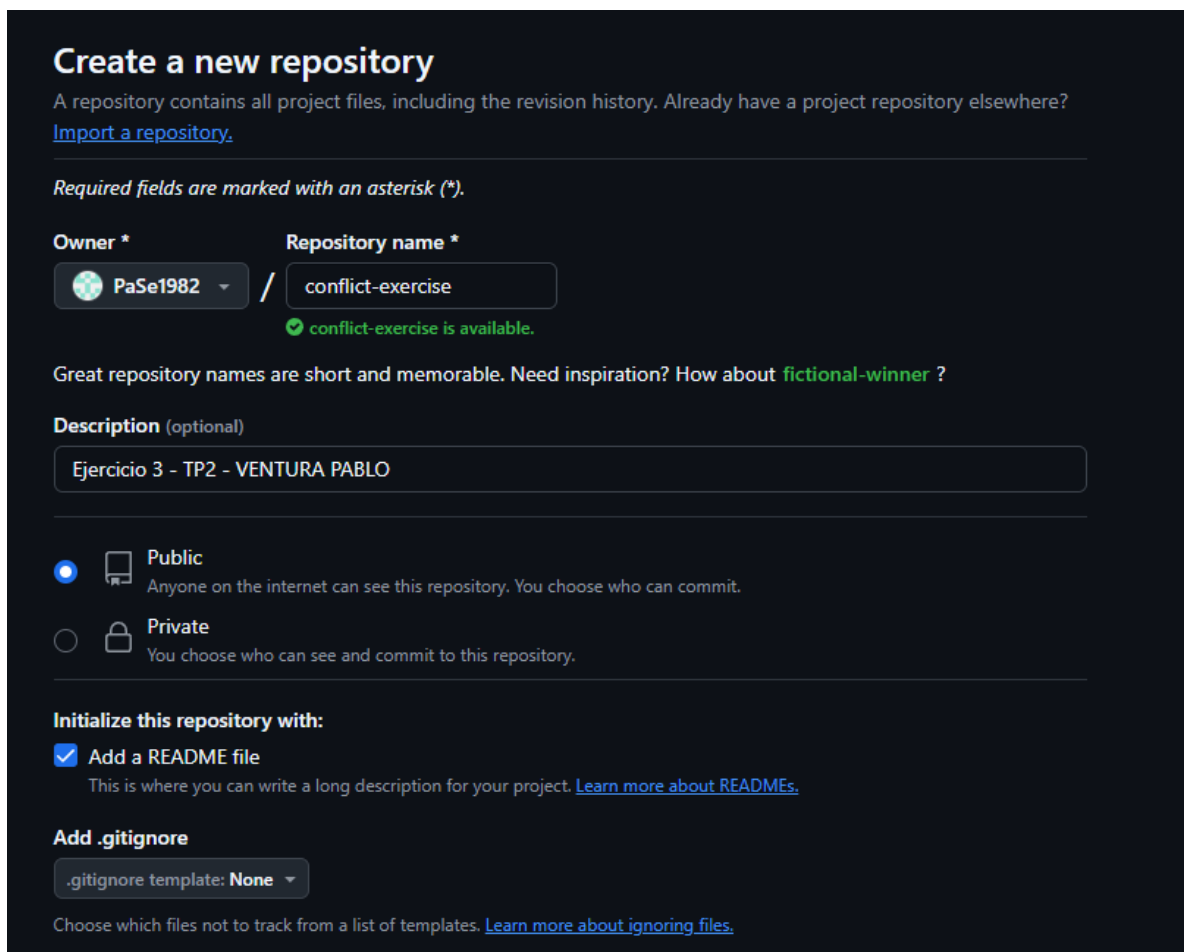
Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/ejercicio2_repositorio (ramal)
$

```


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".




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 ***


 PaSe1982 / conflict-exercise


✔ conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about **fictional-winner** ?

Description (optional)

Ejercicio 3 - TP2 - VENTURA PABLO

☒  **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.](#)

Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio: `https://github.com/PaSe1982/conflict-exercise.git`
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando:
`git clone https://github.com/ PaSe1982/conflict-exercise.git`
- Entra en el directorio del repositorio:
`cd conflict-exercise`

```
MINGW64:/c/Users/Pablo/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise

Pablo@DESKTOP-01PL0T7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION
I (master)
$ git clone https://github.com/PaSe1982/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.

Pablo@DESKTOP-01PL0T7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I (master)
$ cd conflict-exercise

Pablo@DESKTOP-01PL0T7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main)
$ |
```

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:
`git checkout -b feature-branch`

```
MINGW64:/c/Users/Pablo/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise

Pablo@DESKTOP-01PL0T7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

Pablo@DESKTOP-01PL0T7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (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"
```

```

MINGW64:/c/Users/Pablo/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise
Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (feature-branch)
$ git add .

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (feature-branch)
$ git commit -m "Se agregó línea en readme"
[feature-branch d3db449] Se agregó línea en readme
1 file changed, 1 insertion(+)

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (feature-branch)
$ |

```

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"

```

MINGW64:/c/Users/Pablo/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise
Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main)
$ git add .

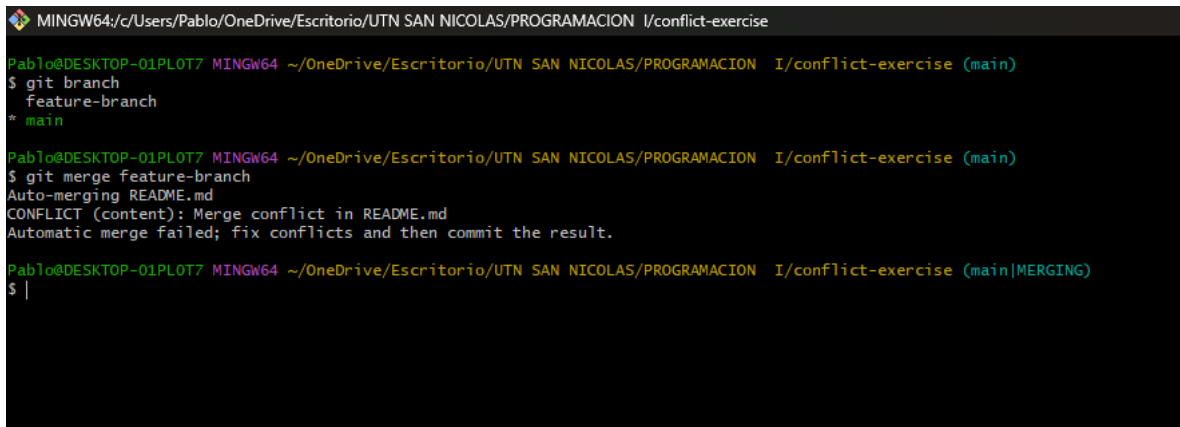
Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main)
$ git commit -m "Línea agregada a la main branch"
[main 1ce28fc] Línea agregada a la main branch
1 file changed, 2 insertions(+)

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main)
$ |

```

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.



```

MINGW64/c/Users/Pablo/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise
Pablo@DESKTOP-01PL0T7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main)
$ git branch
  feature-branch
* main

Pablo@DESKTOP-01PL0T7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main)
$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

Pablo@DESKTOP-01PL0T7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main|MERGING)
$ |

```

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.

=====

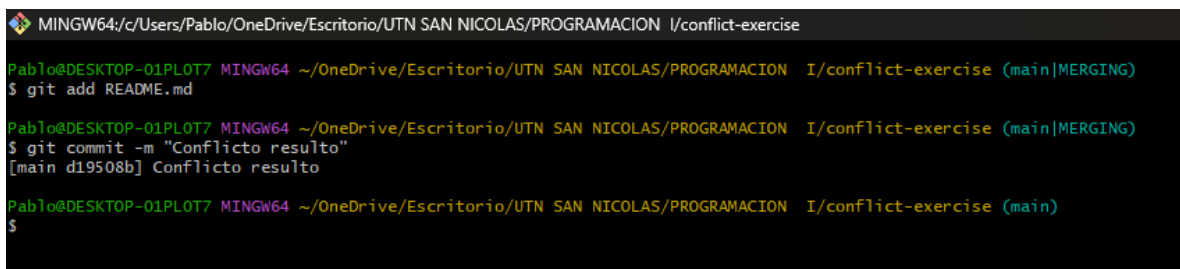
Este es un cambio en la feature branch.

>>>>>>> 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"



```

MINGW64/c/Users/Pablo/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise
Pablo@DESKTOP-01PL0T7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main|MERGING)
$ git add README.md

Pablo@DESKTOP-01PL0T7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main|MERGING)
$ git commit -m "Conflicto resuelto"
[main d19508b] Conflicto resuelto

Pablo@DESKTOP-01PL0T7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main)
$

```

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

```

MINGW64/c/Users/Pablo/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 894 bytes | 894.00 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/PaSe1982/conflict-exercise.git
   3586811..d19508b  main -> main

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (main)
$ git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/PaSe1982/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/PaSe1982/conflict-exercise.git
   * [new branch]      feature-branch -> feature-branch

Pablo@DESKTOP-01PLOT7 MINGW64 ~/OneDrive/Escritorio/UTN SAN NICOLAS/PROGRAMACION I/conflict-exercise (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.

The screenshot shows the GitHub repository page for 'PaSe1982 / conflict-exercise'. The repository is public and has 2 branches and 0 tags. The main branch is selected. The README file is visible, showing the title 'conflict-exercise' and the content 'Ejercicio 3 - TP2 - VENTURA PABLO' and 'Este es un cambio en el feature-branch'. The repository has 4 commits and 0 stars. The right sidebar shows the 'About' section with the title 'Ejercicio 3 - TP2 - VENTURA PABLO' and links to the README, Activity, Stars, Watching, and Forks sections. The 'Releases' section shows 'No releases published' and a link to 'Create a new release'. The 'Packages' section shows 'No packages published' and a link to 'Publish your first package'.