González Federico
34605567

Práctico 2: Git y GitHub

Actividades

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

- ¿Qué es GitHub?

  Es una plataforma de desarrollo colaborativo, que permite al usuario almacenar, compartir y editar código.

- ¿Cómo crear un repositorio en GitHub?

  Para crear un repositorio hay que entrar a la pagina de Github, dirigirse a la parte superior derecha, seleccionar el + y luego hacer click en **Nuevo Repositorio**, escribir un nombre al repositorio y luego dar click en **Crear Repositorio**.

- ¿Cómo crear una rama en Git?

  Para crear una rama, se usa el comando **git branch** ”nombre”.

- ¿Cómo cambiar a una rama en Git?

  Se utiliza el comando **git checkout** “nombre_de_la_rama”.

- ¿Cómo fusionar ramas en Git?

  Se utiliza el comando **git marge** “rama_que_se_quiere_fusionar”.

- ¿Cómo crear un commit en Git?

  Se utiliza el comando **git commit -m** “el texto para identificar porque se hizo el commit”

- ¿Cómo enviar un commit a GitHub?

  Para enviar un commit antes hay que usar los siguientes comandos

  git add . → Añade los cambios realizados

  git commit -m “el texto para identificar el commit” → crea una instantánea de los cambios preparados

  git push origin master → sube los cambios a GitHub.

- ¿Qué es un repositorio remoto?

Es una copia de un proyecto que se encuenta alojada en un servidor remoto.

- ¿Cómo agregar un repositorio remoto a Git?

  Para agregar un repositorio se puede usar el comando **git remote add origin** url.

- ¿Cómo empujar cambios a un repositorio remoto?

  Se utiliza el comando **git push**.

- ¿Cómo tirar de cambios de un repositorio remoto?

  Se utiliza el comando **git pull**.

- ¿Qué es un fork de repositorio?

  Es la copia de un repositorio original.

- ¿Cómo crear un fork de un repositorio?

  Para crear un fork hay que situarse sobre el repositorio que se quiere copiar, en la esquina superior derecha está el botón de **Fork**, y luego se hace click en **crear fork**.

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
  - Primero hay que ir a la página principal del repositorio.
  - Seleccionar la rama que contiene los cambios en el menú "Rama".
  - En el banner amarillo, hacer clic en "Comparar y solicitud de incorporación de cambios".
  - Seleccionar la rama base y la rama de comparación.
  - Escribir un título y una descripción para la solicitud de extracción.
  - Hacer clic en "Crear solicitud de incorporación de cambios".

- ¿Cómo aceptar una solicitud de extracción?

  En la barra de navegación superior, hacer click en la pestaña de **Pull request** (Solicitudes de extracciones), y luego hacer click en el botón **Approve** (aprobar).

- ¿Qué es un etiqueta en Git?

  Es un marcador que se utiliza para marcar puntos específicos en el historial de un repositorio

- ¿Cómo crear una etiqueta en Git?

  Se utiliza el comando **git tag** "nombre de la etiqueta"

- ¿Cómo enviar una etiqueta a GitHub?

  Con el comando **git push origin** --tags

- ¿Qué es un historial de Git?

  Es un registro de los cambios realizados en un repositorio de código.

- ¿Cómo ver el historial de Git?

  Se utiliza el comando **git log**

  - ¿Cómo buscar en el historial de Git?

    Se puede usar el comando **git log** o también buscar por mensaje de commit con el comando **git log –grep** = "nombre del commit"

  - ¿Cómo borrar el historial de Git?

    Se puede borrar, eliminando la carpeta .git local.

- ¿Qué es un repositorio privado en GitHub?

  Es donde se puede guardar código, archivos y revisiones de forma segura y solo visible para el usuario y las cuentas a las que se les otorgue acceso.

- ¿Cómo crear un repositorio privado en GitHub?

  En la parte donde se crea el repositorio habitualmente, antes de presionar el botón de **crear repositorio** hay que elegir la opción de repositorio **Private**.

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

  Dentro del repositorio, en la barra lateral hay que hacer clic en colaboradores, **Agregar personas**.

- ¿Qué es un repositorio público en GitHub?

  Es un espacio virtual donde se puede almacenar y compartir código, archivos y revisiones de forma gratuita y accesible para todos en internet

- ¿Cómo crear un repositorio público en GitHub?

  En la parte donde se crea el repositorio habitualmente, antes de presionar el botón de **crear repositorio** hay que elegir la opción de repositorio **Public**.

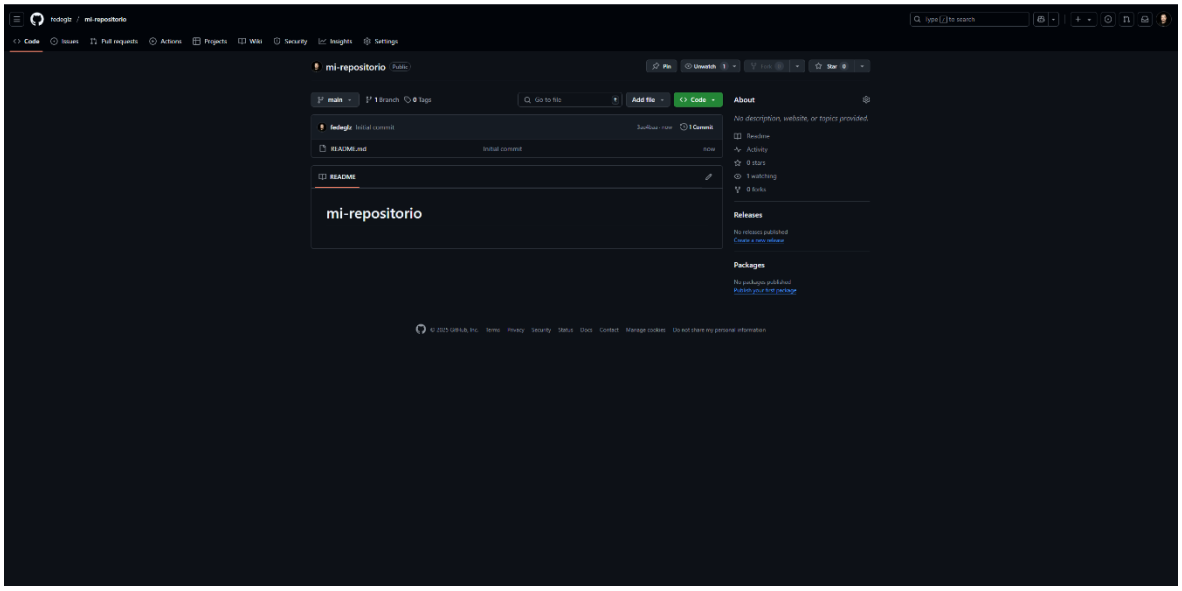
- ¿Cómo compartir un repositorio público en GitHub?

  Dentro del repositorio, en la barra lateral hay que hacer clic en colaboradores, **Agregar personas**.

2) Realizar la siguiente actividad:

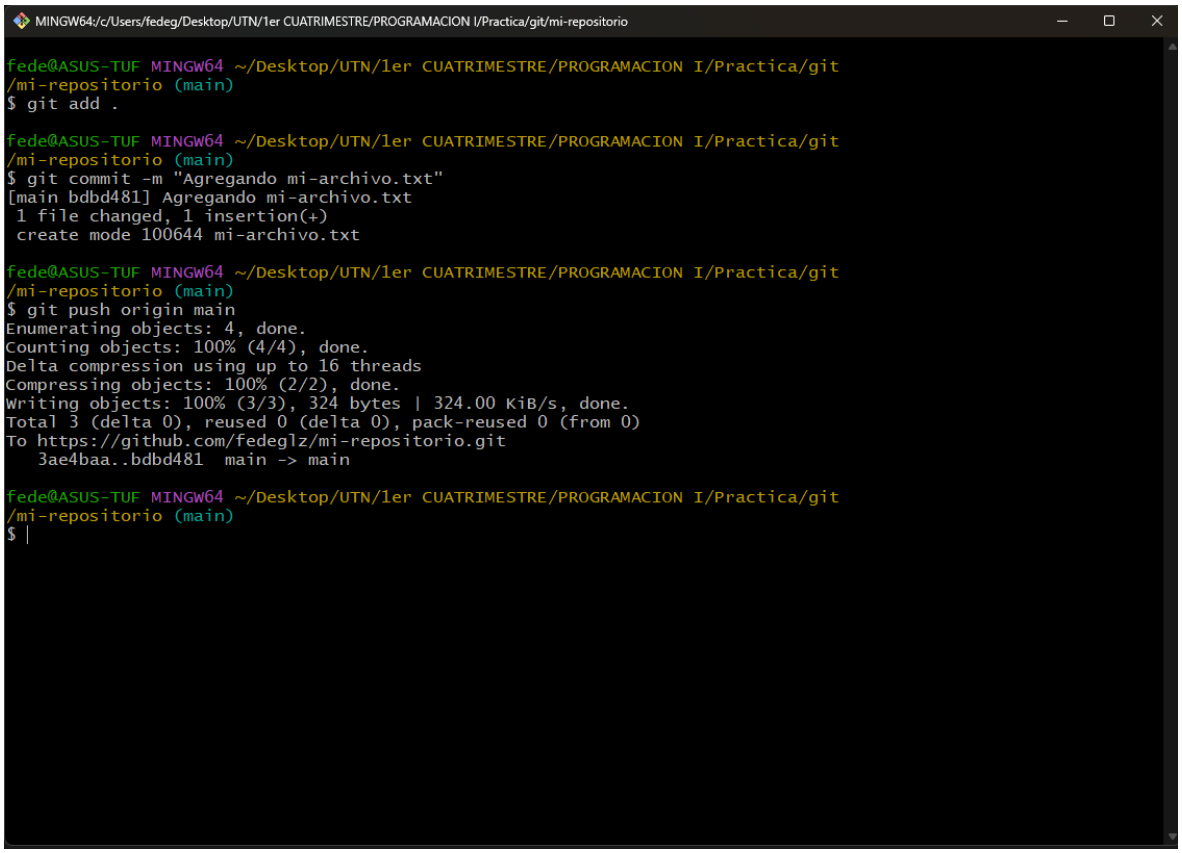
   - Crear un repositorio.

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



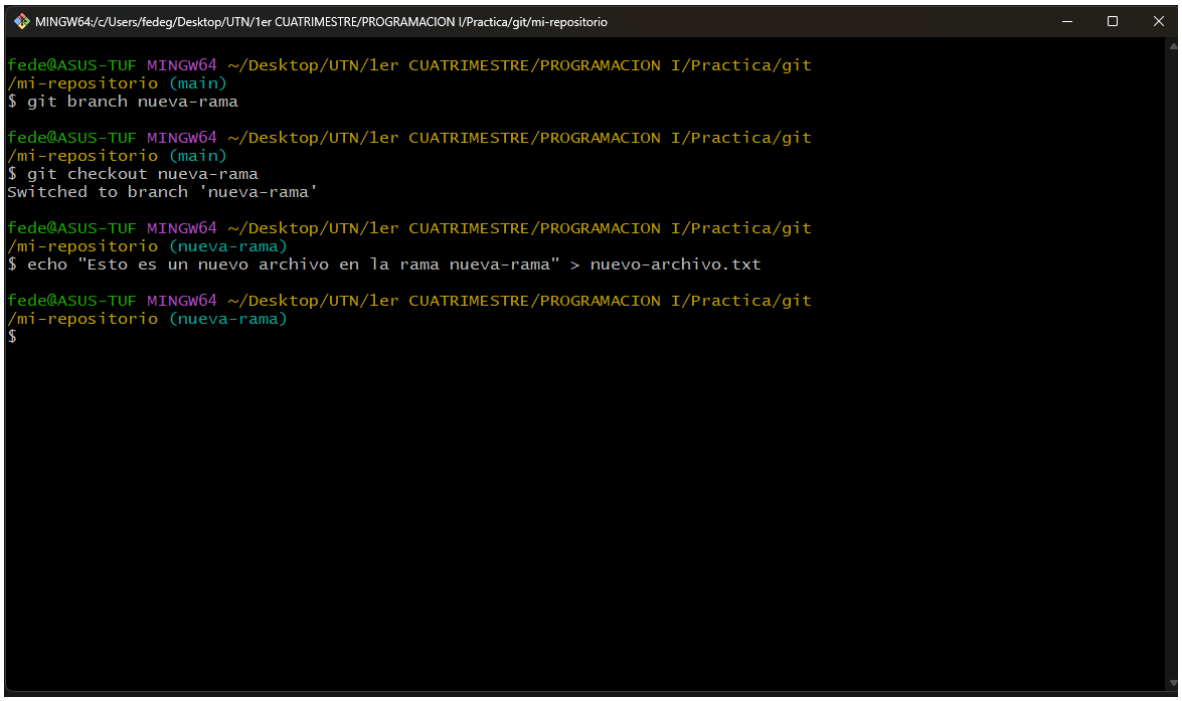
- Agregando un Archivo

    - 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



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
```

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"
```

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.

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"
```

Paso 7: Subir los cambios a GitHub

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

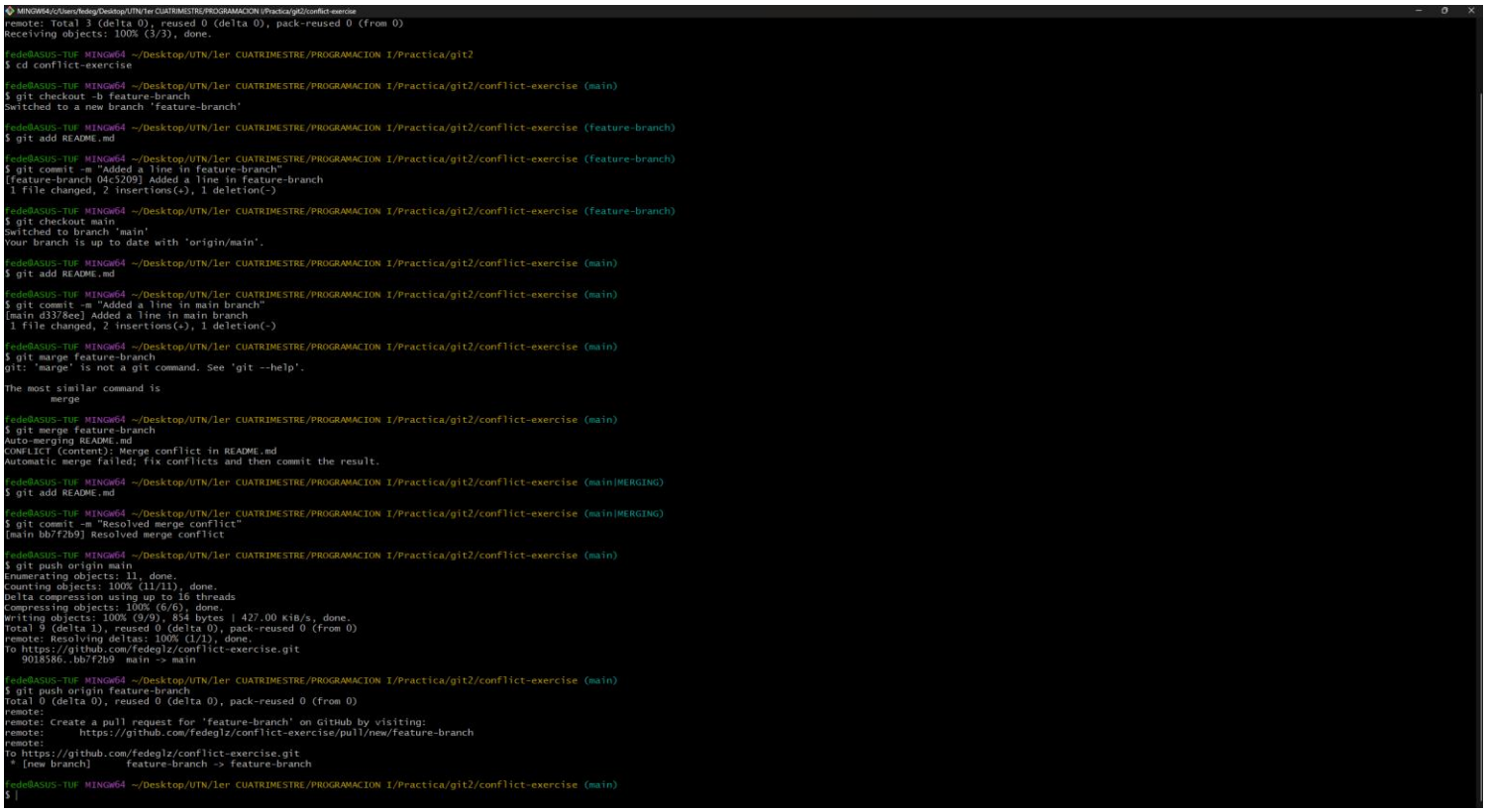
```
git push origin main
```

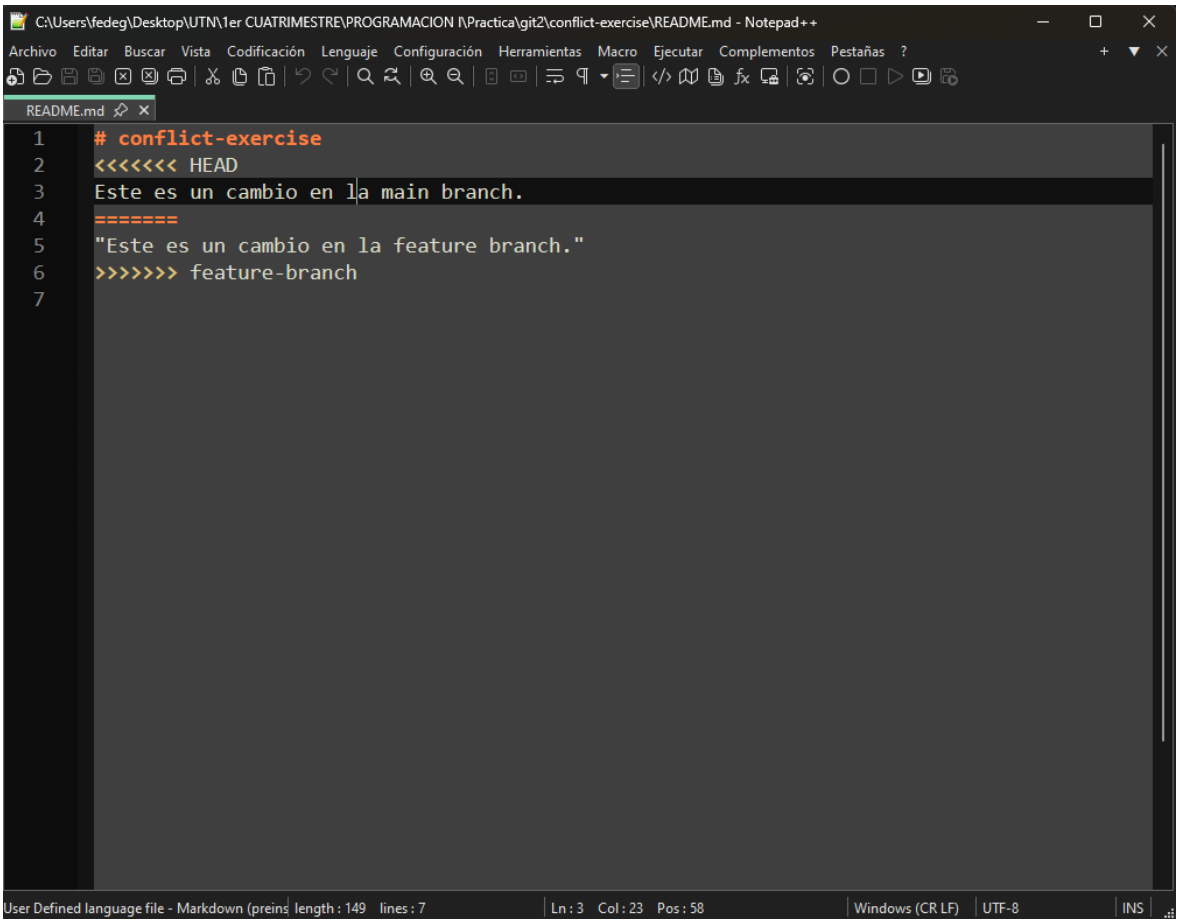
- También sube la feature-branch si deseas:

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

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.


MINGW64:/c/Users/fedeg/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/conflict-exercise
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2
$ cd conflict-exercise
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/conflict-exercise (feature-branch)
$ git add README.md
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 04c5209] Added a line in feature-branch
1 file changed, 2 insertions(+), 1 deletion(-)
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/conflict-exercise (main)
$ git add README.md
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main d337ee] Added a line in main branch
1 file changed, 2 insertions(+), 1 deletion(-)
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/conflict-exercise (main)
$ git marge feature-branch
git: 'marge' is not a git command. See 'git --help'.
The most similar command is
merge
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/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.
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/conflict-exercise (main|MERGING)
$ git add README.md
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main bb7f2b9] Resolved merge conflict
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/conflict-exercise (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), 854 bytes | 427.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/fedeglz/conflict-exercise.git
9018586..bb7f2b9 main -> main
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/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/fedeglz/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/fedeglz/conflict-exercise.git
* [new branch] feature-branch -> feature-branch
Fede@ASUS-TUF MINGW64 ~/Desktop/UTN/1er CUATRIMESTRE/PROGRAMACION I/Practica/git2/conflict-exercise (main)
$


C:\Users\fedeg\Desktop\UTN\1er CUATRIMESTRE\PROGRAMACION I\Practica\git2\conflict-exercise\README.md - Notepad++
Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Complementos Pestañas ?
README.md
1 # conflict-exercise
2 <<<<<<< HEAD
3 Este es un cambio en la main branch.
4 =======
5 "Este es un cambio en la feature branch."
6 >>>>>>> feature-branch
7
User Defined language file - Markdown (preins length : 149 lines : 7
Ln : 3 Col : 23 Pos : 58
Windows (CR LF) UTF-8 INS