Programación I:

Alumno: Martín Eluney Gómez Piñeiro

Trabajo Practico2:

Git y GitHub:

1. Tareas:

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

- ¿Qué es GitHub?
  GitHub es una plataforma para programadores que aloja proyectos usando el sistema de control de versiones Git.
- ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en Github primero se necesita tener una cuenta en la plataforma, luego desde la pantalla inicial una vez registrado, dirigirte al icono de un '+' en la parte superior derecha, hacer click en el y luego en 'New repository', una vez hecho esto se le debe dar un nombre al repositorio, elegir si se desea que el repositorio sea publico o privado y por último, hacer click en el botón 'Create repositoty' debajo de todo.

- ¿Cómo crear una rama en Git?
  Para crear una rama en Git se debe ingresar el siguiente comando por consola: git branch **nombre.**
  Donde **nombre** es el nombre que se le desea poner a la rama.

- ¿Cómo cambiar a una rama en Git?
  Para cambiar de rama se debe ingresar el siguiente comando por consola: git checkout **nombre.**

- ¿Cómo fusionar ramas en Git?
  Para fusionar ramas se debe estar colocado sobre la rama en la que se desea realizar la fusión e ingresar por consola el siguiente comando, donde **nombre** es el nombre de la rama con la que se desea fusionar la rema actual: git merge **nombre.**

- ¿Cómo crear un commit en Git?
  Para crear un commit en git primero se debe ingresar por consola el comando:
  git add .
  El cual almacena todos los cambios realizados hasta el momento, luego se ingresa el comando:
  Git commit -m ‘mensaje’
  Que guarda los cambios junto con un mensaje que preferentemente debe describir cual fue el cambio realizado.

- ¿Cómo enviar un commit a GitHub?
  Para enviar un commit a Github se debe ingresar el siguiente comando por consola:
  Git push -u origin master.

- ¿Qué es un repositorio remoto?
  Un repositorio remoto es una versión de un proyecto de programación alojada en la red, permitiendo ser accedida desde otras computadoras en diferentes ubicaciones.

- ¿Cómo agregar un repositorio remoto a Git?
  Para agregar un nuevo repositorio remoto a Git se debe ingresar por consola el siguiente comando:
  Git remote add origin **url**
  Siendo **url** la dirección web en donde esta alojado el repositorio deseado.

- ¿Cómo empujar cambios a un repositorio remoto?
  Para empujar cambios a un repositorio remoto se debe ingresar por pantalla el siguiente comando por pantalla:
  Git push.

- ¿Cómo tirar de cambios de un repositorio remoto?
  Para tirar de los cambios alojados en el repositoriuo remoto debes ingresar el siguiente comando por pantalla:
  Git pull origin master

- ¿Qué es un fork de repositorio?
  Un fork es una copia realizada por nosotros sobre un repositorio remoto, esto nos permite tener el repositorio y realizar los cambios que desiemos sin alterar el original ni crear una rama.

- ¿Cómo crear un fork de un repositorio?
  Para realizar un fork en un repositorio, debes dirijirte a la url del mismo y hacer clik sobre el botón fork que se encuentra en la parte superior derecha, nombrar el fork deseado y darle al botón créate fork abajo a la derecha.

• ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
Primero debes haber realizado un fork y haber realizado cambios en este, una vez hecho esto deber hacer click en el botón ‘New pull request’.

• ¿Cómo aceptar una solicitud de extracción?
Para aceptar una extracción debes estar ubicado en tu repositorio, ira a la pestaña ‘Pull request’, seleccionar la solicitud deseada y hacer click en ‘Merge pull request’

• ¿Qué es una etiqueta en Git?
Una etiqueta en Git es un marcador que se utiliza para señalar versiones especificas.

• ¿Cómo crear una etiqueta en Git?
Para crear una etiqueta en Git debes ingresar por consola el siguiente comando:
Git tag v1.0

• ¿Cómo enviar una etiqueta a GitHub?
Para enviar una etiqueta a github debes ingresar por consola el siguiente comando:
git push origin v1.0

• ¿Qué es un historial de Git?
Un historial de Git es el lugar donde se registran todos los cambios realizados en un proyecto.

• ¿Cómo ver el historial de Git?
Para ver el historial de Git debes ingresar por consola el siguiente comando:
git log

• ¿Cómo buscar en el historial de Git?
Para buscar en el historial debes ingresar el siguiente comando por consola:
git log --grep="**palabra_clave**"

• ¿Cómo borrar el historial de Git?
En la documentación de Git no hay ninguna forma para eliminar el hitorial del mismo

• ¿Qué es un repositorio privado en GitHub?
Un repositorio privado de github es un repositorio el cual no puede ser visto por cualquier persona solo aquellos a los que se les haya dado acceso.

• ¿Cómo crear un repositorio privado en GitHub?
Al momento de crear un nuevo repositorio se debe marcar la casilla que dice private.

• ¿Cómo invitar a alguien a un repositorio privado en GitHub?
Para invitar a alguien a un repositorio privado de GitHub se debe ingresar a ‘settings’, luego a ‘Collabotators and teams’ y por últimos ‘Add people’.

• ¿Qué es un repositorio público en GitHub?
Es un repositorio visible para todos

• ¿Cómo crear un repositorio público en GitHub?
Al momento de crear un nuevo repositorio se debe marcar la casilla que dice public.

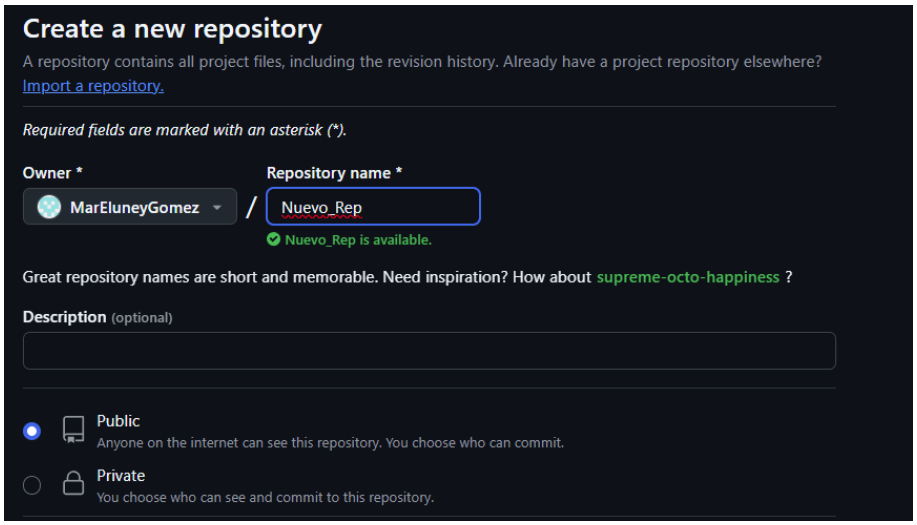
• ¿Cómo compartir un repositorio público en GitHub?
Tan solo debes compartir la url del mismo.

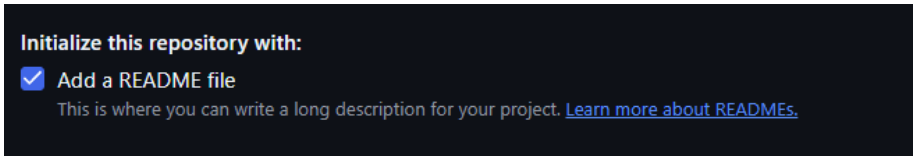
Ejercicio 2:

Realizar la siguiente actividad:

Crear un repositorio.

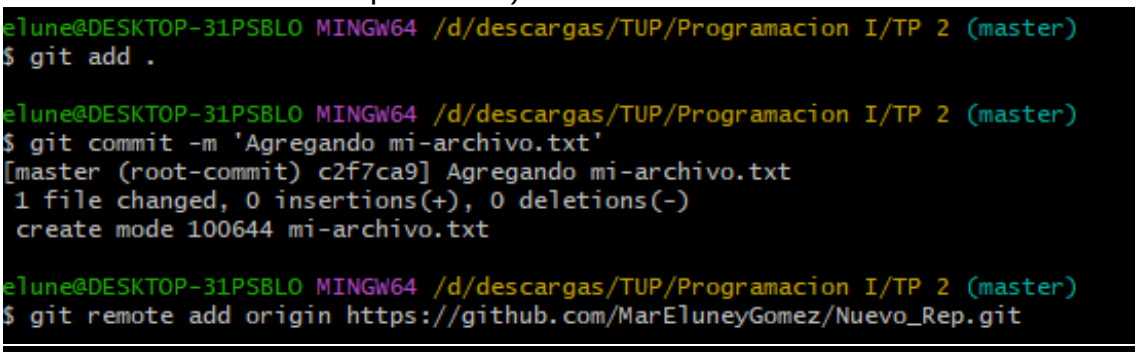
- Dale un nombre al repositorio.
- Elije el repositorio sea público.
- Inicializa el repositorio con un archivo.

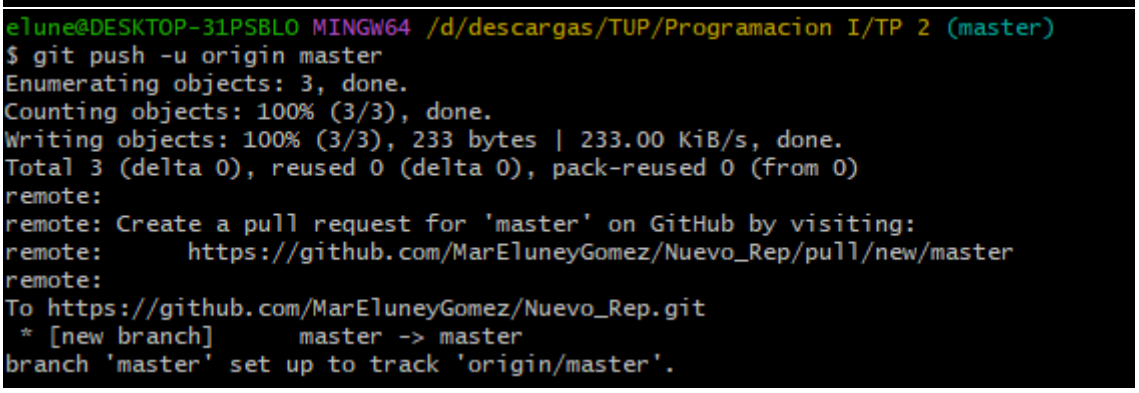




Agregando un Archivo

- 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

```
elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 (master)
$ git branch rama2

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 (master)
$ git branch
* master
  rama2

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 (master)
$ git checkout rama2
Switched to branch 'rama2'

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 (rama2)
$ git add .

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 (rama2)
$ git comit -m 'un cambio'
git: 'comit' is not a git command. See 'git --help'.

The most similar command is
        commit

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 (rama2)
$ git commit -m 'un cambio'
[rama2 a51cbac] un cambio
 1 file changed, 1 insertion(+)

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 (rama2)
$ git push -u origin rama2
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 262 bytes | 262.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'rama2' on GitHub by visiting:
remote:      https://github.com/MarEluneyGomez/Nuevo_Rep/pull/new/rama2
remote:
To https://github.com/MarEluneyGomez/Nuevo_Rep.git
 * [new branch]      rama2 -> rama2
branch 'rama2' set up to track 'origin/rama2'.

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 (rama2)
$
```

Ejercicio 3:

Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

• Ve a GitHub e inicia sesión en tu cuenta.

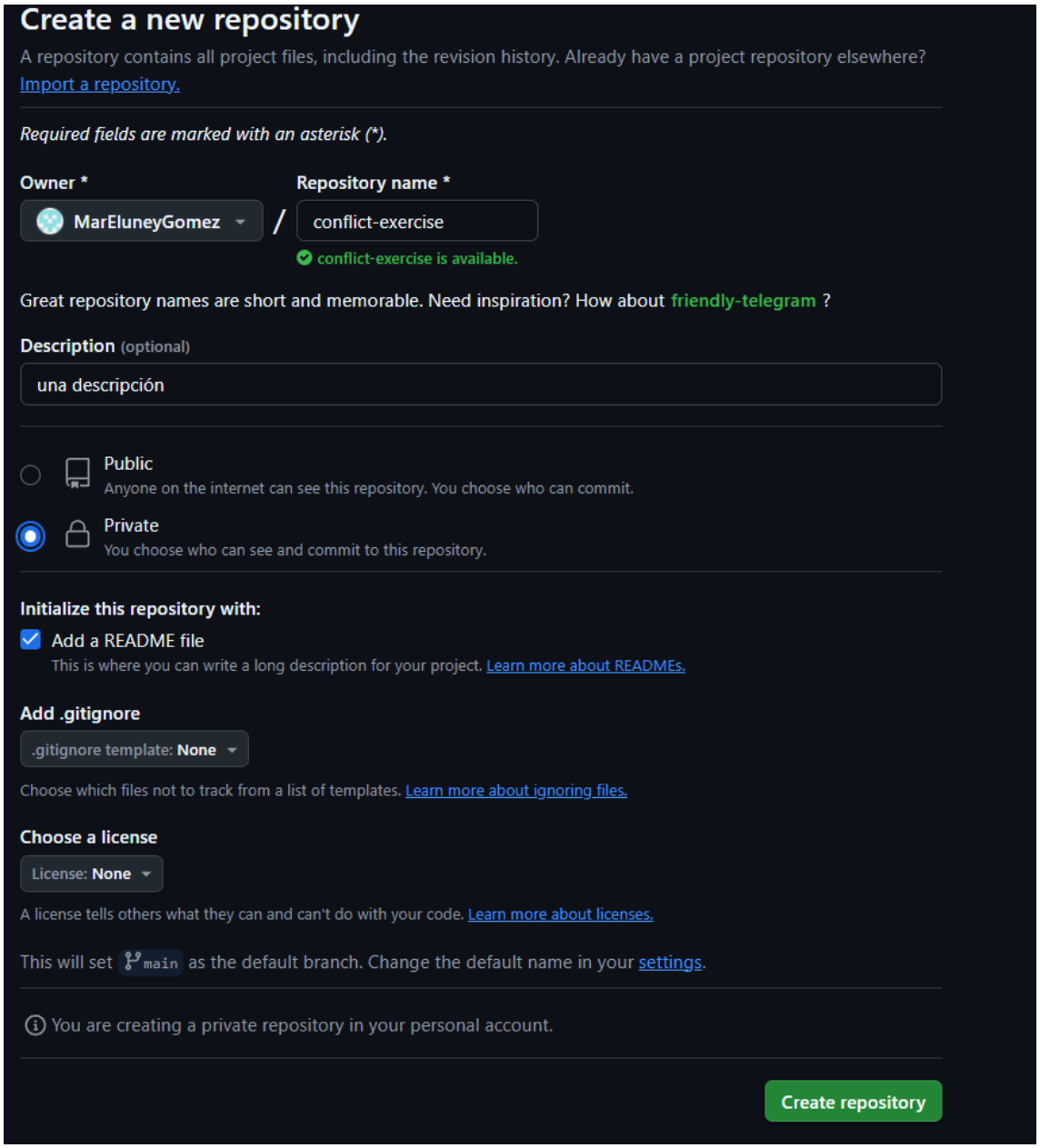
• Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.

• 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

```
elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3
$ git clone https://github.com/MarEluneyGomez/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3
$ cd conflict-exercise

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/confli
ct-exercise (main)
$
```

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"

```
elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/confli
ct-exercise (main)
$ git branch feature-branch

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/conflict-exercise (main)
$ git checkout -b feature-branch
fatal: a branch named 'feature-branch' already exists

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/conflict-exercise (main)
$ git checkout feature-branch
Switched to branch 'feature-branch'

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/conflict-exercise (feature-branch)
$ git add README.md

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 5e06632] 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

• 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"

```
elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/conflict-exercise (main)
$ git checkout feature-branch
Switched to branch 'feature-branch'

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/conflict-exercise (feature
$ git add README.md

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/conflict-exercise (feature
$ git commit -m "Added a line in feature-branch"
[feature-branch 5e06632] Added a line in feature-branch
 1 file changed, 1 insertion(+)

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/conflict-exercise (feature
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/conflict-exercise (main)
$ git add README.md
git commit -m "Added a line in main branch"
[main 86573e0] Added a line in main branch
 1 file changed, 1 insertion(+)
```

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.

```
elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/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.
```

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

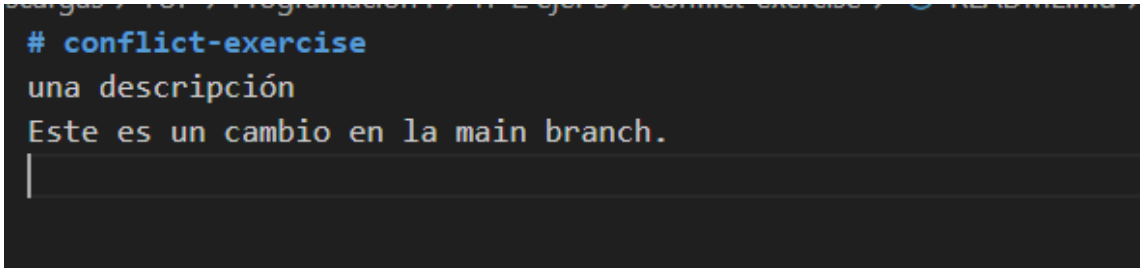
• Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.

• 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"




```
elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/conflict-exercise (main|MERGING)
$ git add README.md

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/conflict-exercise (main|MERGING)
$ git commit -m 'resolved merge conflict'
[main 25ca115] 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

```
elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/conflict-exercise (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 6 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 828 bytes | 414.00 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/MarEluneyGomez/conflict-exercise.git
   79f7956..25ca115  main -> main

elune@DESKTOP-31PSBLO MINGW64 /d/descargas/TUP/Programacion I/TP 2 ejer 3/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/MarEluneyGomez/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/MarEluneyGomez/conflict-exercise.git
 * [new branch]      feature-branch -> 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.