



TECNICATURA EN PROGRAMACION
A DISTANCA

TRABAJO PRACTICO N-2

GIT GITHUB

Materia Programacion I

Docente: Ariel Enferrel

Tutor: Franco Gonzalez

Alumna : Yohanna Díaz Monroy

Actividades

1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada

(Desarrollar las respuestas):

a) ¿Qué es GitHub?

Es una comunidad donde se puede compartir, nuestros repositorios de Git de forma publica o privada
Un espacio en línea donde los desarrolladores pueden almacenar, gestionar y colaborar en proyectos de código.

b) ¿Cómo crear un repositorio en GitHub?

Creada una cuenta en GitHub, esta sera simil red social donde podremos crear un repositorio.

Desde ahi debemos seguir unos pasos

1. Inicia sesión en GitHub en mi cuenta
2. Navega a la página de creación de repositorios
3. En la esquina superior derecha de la pagina de GitHub, hacer click en el signo +
4. selecciona nuevo repositorio
5. Elegir un nombre
6. Eligir si es público o privado
7. Crear repositorio

c) ¿Cómo crear una rama en Git?

Para crear una nueva rama utilizamos el comando **git branch**

Nombre de la rama que queremos darle

d) ¿Cómo cambiar a una rama en Git?

Para cambiar de rama ejecutamos el comando **git checkout** y el nombre de la rama

e) ¿Cómo fusionar ramas en Git?

Para fusionar ramas se usa el comando **git merge** seguido del nombre de la rama que se desea fusionar

f) ¿Cómo crear un commit en Git?

Para crear un commit usamos el comando **git commit** la opción **-m**, **git commit -m** y el "mensaje"

g) ¿Cómo enviar un commit a GitHub?

Para enviar un commit a Github se utiliza el comando **git push** y nombre_de_la_rama

h) ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión del proyecto que se almacena en un servidor en línea.

i) ¿Cómo agregar un repositorio remoto a Git?

Utiliza el comando `git remote add` para vincular tu repositorio local con un repositorio remoto y asignar un nombre a ese remoto:

git remote add [nombre] [url]

j) ¿Cómo empujar cambios a un repositorio remoto?

Para empujar cambios al repositorio remoto utilizamos el comando **git push**, que se utiliza para enviar tus commits locales al repositorio remoto

git push origin y nombre de la rama

k) ¿Cómo tirar de cambios de un repositorio remoto?

Se pueden utilizar dos comandos pero tienen a su vez utilidades diferentes.

git fetch: Descarga los cambios pero no los combina.

git fetch origin y nombre de la rama

git pull: Descarga y combina los cambios automáticamente.

git pull origin y nombre de la rama

l) ¿Qué es un fork de repositorio?

Un fork es la creación de una copia personal de un repositorio existente. Esta copia se almacena en la cuenta personal y permite realizar modificaciones y experimentar con el código sin afectar el repositorio original.

m) ¿Cómo crear un fork de un repositorio?

1. Ir a la página del repositorio del cual se desea realizar el fork .
2. Hacer clic en el botón Fork, En la esquina superior derecha de la pagina
3. Selecciona mi cuenta
4. GitHub creará una copia del repositorio en mi cuenta.

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

Con este proceso se proponen los cambios que se han realizado desde el fork o rama a la rama principal del repositorio original. Vamos a la solapa pull request de ahí clic en el botón "New pull request" Escribe un título y una descripción detallada de los cambios que se han realizado y porque se los considera necesarios. Luego daremos clic en "Create pull request".

o) ¿Cómo aceptar una solicitud de extracción?

En el proceso en el cual el autor acepta las modificaciones que se le enviaron el pull request y si esta de acuerdo y las acepta las incluire en la rama maestra. La forma de hacerlo es hacer click en **Merge pull request**.

p) ¿Qué es un etiqueta en Git?

Una etiqueta en Git es una forma de marcar un punto específico en el historial del repositorio, un marcador permanente que apunta a un commit concreto. Utilizado por lo general para marcar las versiones de lanzamiento ejemplo : v1.1

q) ¿Cómo crear una etiqueta en Git?

Git posee dos tipos de etiquetas

- Etiquetas ligeras: Son simplemente punteros a un commit específico.
- Etiquetas anotadas: Almacenan información adicional, como el nombre del etiquetador, la fecha y un mensaje. Se recomiendan para versiones de lanzamiento públicas.

Crear una etiqueta anotada:

git tag -a y nombre que se le dara a la etiqueta -m "descripcion"

Ejemplo: **git tag -a** v1.1 -m "Versión de lanzamiento 1.1"

Crear una etiqueta ligera:

git tag y nombre que se le dara a la etiqueta

Ejemplo: **git tag** v1.1-ligera

r) ¿Cómo enviar una etiqueta a GitHub?

Tener creada previamente una etiqueta en el repositorio local, ligera o anotada, verificar que exista con el comando **git tag**.

Para enviar una etiqueta se usa el comando: **git push** origin nombre de la etiqueta

Ejemplo: **git push** origin v1.1

s) ¿Qué es un historial de Git?

Es el registro cronológico de todos los cambios realizados en un repositorio a lo largo del tiempo, permite rastrear y gestionar las modificaciones del código fuente. Esto proporciona una visión clara de cómo se ha desarrollado el proyecto y cómo se han integrado los cambios. Se visualiza con el comando **git log**

t) ¿Cómo ver el historial de Git?

En el historial se pueden ver los Commits, que son instantáneas de los cambios realizados en el repositorio en un momento dado, información como el autor, la fecha, la hora y un mensaje que describe los cambios. Permite rastrear quién realizó qué cambios, cuándo y por qué. Permite también poder volver a versiones anteriores del código si es necesario. y entre otros en el historial también registra la creación y fusión de ramas.

u) ¿Cómo buscar en el historial de Git?

En el historial de Git se puede rastrear cambios, encontrar versiones específicas y comprender la evolución de un proyecto. El comando es **git log**.

v) ¿Cómo borrar el historial de Git?

Se debe tener mucho cuidado al momento de eliminar información del historial, ya que puede afectar todo el Proyecto, de igual forma muchas veces es necesario realizar cambio. El comando para realizar borrado es **git reset**

w) ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es el espacio de almacenamiento en línea para los código que solo serán accesible para el Usuario y para las personas que este le de acceso.

x) ¿Cómo crear un repositorio privado en GitHub?

En la cuenta de GitHub

- En la esquina superior derecha clic en el botón "+" selecciona "New repository"
- Dar un nombre al Repositorio
- Descripción breve del propósito del mismo
- Elegir la opción privado para que página sepa a quien mostrarlo

y) ¿Cómo invitar a alguien a un repositorio privado en GitHub?

- Acceder al repositorio
- Hacer clic en la pestaña **Settings**
- Seleccionar **Collaborators**
 - Dentro de este en **Add people** y ingresar el nombre del Usuario con el que se desea compartir información.
- Seleccionar el nivel de acceso que le daremos, **Read, Triage, Write, Maintain** o **Admin**.
- Hacer clic en **Add** para enviar la invitación

z) ¿Qué es un repositorio público en GitHub?

Es un espacio de almacenamiento en línea para código y archivos que está accesible para todo público, este tendrá visibilidad global: Cualquier persona puede ver, colaborar, clonar y bifurcar el código en un repositorio público.

aa) ¿Cómo crear un repositorio público en GitHub?

1. Iniciar sesión en GitHub
2. Ingresar a la página de creación de repositorios
3. Esquina superior derecha click en +
4. Seleccionar **New Repository**
5. Click en **New**
6. Darle un nombre
7. Configuración de privacidad elegir **público**
8. Ir al botón **Create Repository**

bb) ¿Cómo compartir un repositorio público en GitHub?

La forma más sencilla es compartir el URL a través de correo electrónico, mensajería instantánea, redes sociales o cualquier otro medio digital.

2) Realizar la siguiente actividad:

- Crear un repositorio.
- Dale un nombre al repositorio.
- Elije el repositorio sea público.

The screenshot shows the GitHub 'New repository' page. Red annotations highlight the following elements:

- The top navigation bar with the 'New repository' link.
- The 'Repository name' field, which contains 'miarchivo1' and has a confirmation message 'miarchivo1 is available'.
- The 'Description (optional)' field, which contains 'practico 2'.
- The 'Public' radio button, which is selected.
- The 'Create repository' button at the bottom right.

The page content includes:

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 *
YOHVASS

Repository name *
miarchivo1
miarchivo1 is available.

Great repository names are short and memorable. Need inspiration? How about [supreme-invention](#) ?

Description (optional)
practico 2

☒ **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.](#)

① You are creating a public repository in your personal account.

[Create repository](#)

<https://github.com/new> 1/2

- 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).

\$ git init

Reinitialized existing Git repository in C:/Users/Usuario/Desktop/ejercicio git/.git/

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (master)

\$ git add .

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (master)

\$ git commit -m "mi primer proyecto"

[master (root-commit) 592ef44] mi primer proyecto

1 file changed, 1 insertion(+)

create mode 100644 mi archivo.txt

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (master)

\$ git remote add origin https://github.com/YOHVASS/miarchivo.git

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (master)

\$ git push origin master

Creando Branchs

- Crear una Branch
- Realizar cambios o agregar un archivo

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (master)

\$ git branch

* master

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (master)

\$ git branch rama1

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (master)

\$ git branch

* master

rama1

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (master)

\$ git branch

* master

rama1


```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (master)
$ git checkout rama1
M      mi archivo.txt
Switched to branch 'rama1'
```

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (rama1)
$ git branch
  master
* rama1
```

- **Subir la Branch**

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (rama1)
$ git checkout master
Switched to branch 'master'
```

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (master)
$ git merge rama1 master
Updating 592ef44..e6d5654
Fast-forward
  mi archivo.txt | 3 ++-
  texto. txt     | 0
  texto2. txt    | 0
3 files changed, 2 insertions(+), 1 deletion(-)
create mode 100644 texto. txt
create mode 100644 texto2. txt
```

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (master)
$ git add .
```

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/ejercicio git (master)
$ git commit -m "cambios master"
On branch master
nothing to commit, working tree clean
```






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".

29/3/25 11:17 New repository


  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 *

 YOHVASS ▾

Repository name *


conflict-exercise


 conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about [didactic-giggle](#) ?

Description (optional)

ejercicio conflicto

☒  **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

<https://github.com/new> 1/2

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`
- Entra en el directorio del repositorio: `cd conflict-exercise`

```

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-
exercise1/conflict-exercise1
$ git clone https://github.com/YOHVASS/conflict-exercise1.git
Cloning into 'conflict-exercise1'...
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.

```

```

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-
exercise1/conflict-exercise1
$ cd conflict-exercise1

```

```

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-
exercise1/conflict-exercise1/conflict-exercise1 (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"`

```

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-
exercise1/conflict-exercise1/conflict-exercise1 (main)
$ git branch feature-branch

```

```

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-
exercise1/conflict-exercise1/conflict-exercise1 (main)
$ git add .

```

```

Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-
exercise1/conflict-exercise1/conflict-exercise1 (main)
$ 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"`

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-exercise1/conflict-exercise1 (main)
$ git branch
  feature-branch
* main
```

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-exercise1/conflict-exercise1 (main)
$ git add .
```

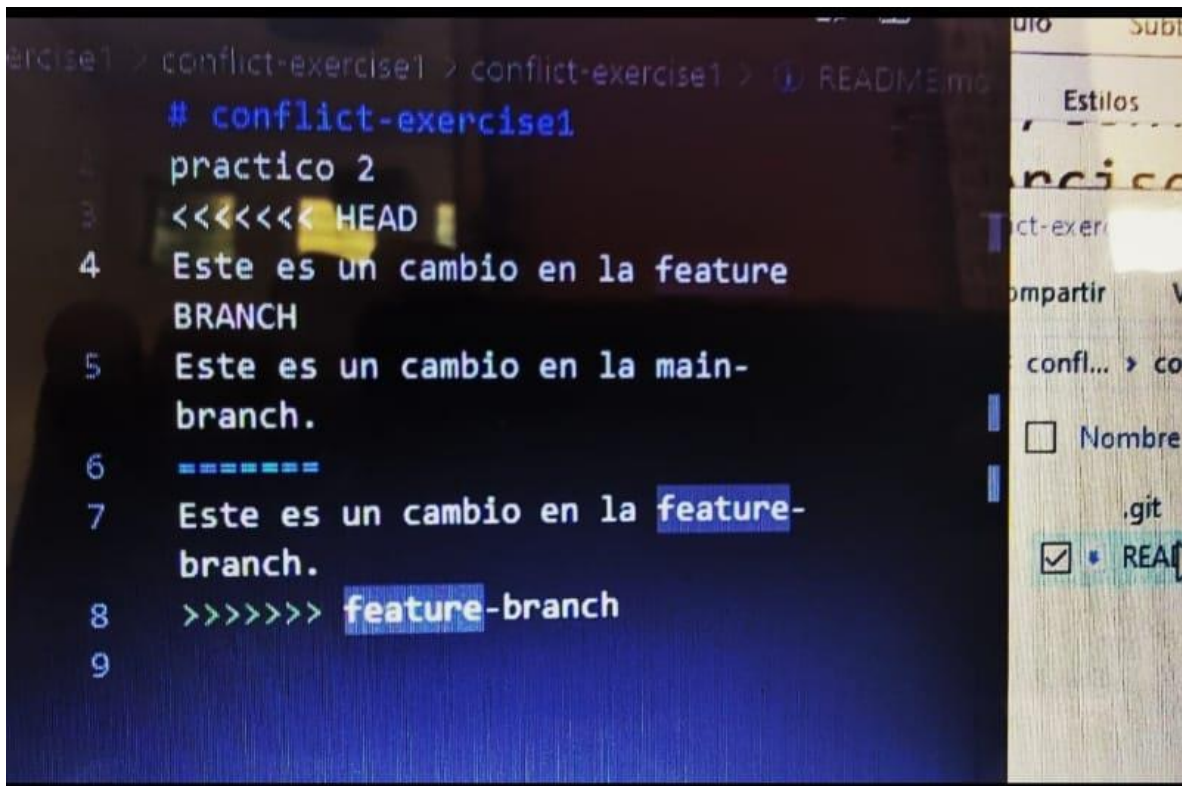
```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-exercise1/conflict-exercise1 (main)
$ git commit -m "Added a line in feature-branch"
On branch main
Your branch is up to date with 'origin/main'.
```

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-exercise1/conflict-exercise1 (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.

```
$ 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.
```



```
conflict-exercise1 > conflict-exercise1 > 1 README.md
# conflict-exercise1
2 practico 2
3 <<<<<<< HEAD
4 Este es un cambio en la feature BRANCH
5 Este es un cambio en la main-branch.
6 =====
7 Este es un cambio en la feature-branch.
8 >>>>>>> feature-branch
9
```

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"`

Se editó el archivo borrando lo marcado en verde

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-exercise1/conflict-exercise1 (main|MERGING)
$ git add .
```

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-exercise1/conflict-exercise1 (main|MERGING)
$ git commit -m "Resolved merge conflict"
```

[main 0826997] 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`

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-  
exercise1/conflict-exercise1/conflict-exercise1 (main)  
$ git push origin main  
info: please complete authentication in your browser...  
Enumerating objects: 14, done.  
Counting objects: 100% (14/14), done.  
Delta compression using up to 2 threads  
Compressing objects: 100% (8/8), done.  
Writing objects: 100% (12/12), 1.06 KiB | 90.00 KiB/s, done.  
Total 12 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)  
remote: Resolving deltas: 100% (3/3), done.  
To https://github.com/YOHVASS/conflict-exercise1.git  
d8e6da5..0826997  main -> main
```

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-  
exercise1/conflict-exercise1/conflict-exercise1 (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.

<https://github.com/YOHVASS/conflict-exercise1/blob/main/README.md>

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-  
exercise1/conflict-exercise1/conflict-exercise1 (main)  
$ git push origin main  
info: please complete authentication in your browser...  
Enumerating objects: 14, done.  
Counting objects: 100% (14/14), done.  
Delta compression using up to 2 threads  
Compressing objects: 100% (8/8), done.  
Writing objects: 100% (12/12), 1.06 KiB | 90.00 KiB/s, done.  
Total 12 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)  
remote: Resolving deltas: 100% (3/3), done.  
To https://github.com/YOHVASS/conflict-exercise1.git  
d8e6da5..0826997  main -> main
```

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-  
exercise1/conflict-exercise1/conflict-exercise1 (main)  
$
```

- Puedes revisar el historial de commits para ver el conflicto y su resolución.

```
Usuario@DESKTOP-VDCK76P MINGW64 ~/Desktop/conflict-  
exercise1/conflict-exercise1/conflict-exercise1 (main)  
$ git log  
commit 08269977c4437116d081c7b77a1aaa39a798556f (HEAD -> main,  
origin/main, orig  
in/HEAD)  
Merge: 6fffd99e 1263dd5  
Author: Yovass <yovassphoto@gmail.com>  
Date: Sat Mar 29 17:41:16 2025 -0300
```

Resolved merge conflict

```
commit 6fffd99eaf9b59097929333e02974b741ae80186c  
Author: Yovass <yovassphoto@gmail.com>  
Date: Sat Mar 29 17:33:39 2025 -0300
```

Added a line in main branch

```
commit 1263dd53ce3d0f73e380752a769eb07b9b2ef20b (feature-branch)  
Author: Yovass <yovassphoto@gmail.com>  
Date: Sat Mar 29 17:31:30 2025 -0300
```

Added a line in feature-branch