

PROGRAMACIÓN 1

TRABAJO PRACTICO 2: GIT Y GITHUB

Actividades:

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

- **¿Qué es GitHub?**

GitHub es una plataforma web para alojar y colaborar en proyectos de software. Utiliza Git como sistema de control de versiones, lo que permite a los desarrolladores gestionar y seguir los cambios en el código, colaborar de forma remota y mantener un historial detallado de las modificaciones.

- **¿Cómo crear un repositorio en GitHub?**

Inicia sesión en tu cuenta de GitHub. -> Haz clic en el botón “New” o “Nuevo repositorio” -> Rellena el nombre del repositorio, agrega una descripción si lo deseas y elige si será público o privado -> Haz clic en “Create repository”.

- **¿Cómo crear una rama en Git?**

Desde la terminal, puedes crear una rama utilizando el comando:

```
$ git branch nombre-de-la-rama
```

- **¿Cómo cambiar a una rama en Git?**

Para cambiar a otra rama, utiliza el comando:

```
$ git checkout nombre-de-la-rama
```

- **¿Cómo fusionar ramas en Git?**

Para fusionar una rama en la rama actual, primero asegúrate de estar en la rama de destino (por ejemplo, main) y luego ejecuta:

```
$ git merge nombre-de-la-rama
```

Si hay conflictos, Git te indicará qué archivos requieren resolución.

- **¿Cómo crear un commit en Git?**

Después de hacer cambios en el código, sigue estos pasos:

```
# Añade los cambios al área de preparación:
```

```
$ git add .
```

```
# Crea el commit con un mensaje descriptivo:
```

```
$ git commit -m "Descripción de los cambios"
```

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

Una vez creado el commit, se puede enviar al repositorio remoto con:

```
$ git push origin nombre-de-la-rama
```

Si estás en la rama principal y deseas actualizarla:

```
$ git push origin main
```

- **¿Qué es un repositorio remoto?**

Un repositorio remoto es una versión del proyecto alojada en un servidor o servicio en la nube (como GitHub, GitLab o Bitbucket). Permite que varios desarrolladores colaboren y mantengan una copia centralizada del código.

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

Para vincular tu repositorio local con uno remoto, utiliza:

```
$ git remote add origin https://github.com/tu-usuario/nombre-del-repositorio.git
```

Esto asigna la URL remota con el alias "origin"

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

Después de realizar commits locales, utiliza el comando:

```
$ git push origin nombre-de-la-rama
```

Este comando envía los commits a la rama correspondiente en el repositorio remoto.

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

Para actualizar tu repositorio local con los cambios del repositorio remoto, utiliza:

```
$ git pull origin nombre-de-la-rama
```

Este comando descarga y fusiona los cambios de la rama remota en tu rama actual.

- **¿Qué es un fork de repositorio?**

Un fork es una copia completa de un repositorio que se realiza en tu cuenta de GitHub. Permite experimentar y hacer cambios sin afectar el proyecto original.

- **¿Cómo crear un fork de un repositorio?**

Para forquear un repositorio, simplemente visita el repositorio original en GitHub y haz clic en el botón "Fork" (generalmente en la esquina superior derecha). GitHub creará una copia del repositorio en tu cuenta.

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

Haz un fork y crea una rama con tus cambios en tu repositorio -> Envía tus commits a tu fork -> En GitHub, navega hasta la página de tu fork y haz clic en "New pull request" -> Compara la rama con la rama base del repositorio original y escribe una descripción clara de tus cambios -> Envía la solicitud de extracción.

- **¿Cómo aceptar una solicitud de extracción?**

El propietario o mantenedor del repositorio revisa la solicitud y, si los cambios son aceptables, procede a hacer merge de la rama del pull request en la rama principal.

Esto se puede realizar directamente desde la interfaz web de GitHub mediante el botón "Merge pull request".

- **¿Qué es un etiqueta en Git?**

Una etiqueta (tag) es una referencia que apunta a un punto específico en la historia del repositorio. Se utiliza comúnmente para marcar versiones de lanzamiento (por ejemplo, v1.0, v2.0).

- **¿Cómo crear una etiqueta en Git?**

Para crear una etiqueta anotada, ejecuta:

```
$ git tag -a v1.0 -m "Versión 1.0"
```

Esto crea una etiqueta llamada "v1.0" con un mensaje descriptivo.

- **¿Cómo enviar una etiqueta a GitHub?**

Después de crear la etiqueta localmente, envíala al repositorio remoto con:

```
$ git push origin --tags
```

Este comando sube todas las etiquetas locales al repositorio remoto.

- **¿Qué es un historial de Git?**

El historial de Git es el registro completo de todos los commits realizados en el repositorio, incluyendo información sobre quién hizo cada cambio, cuándo se hizo y qué se modificó.

- **¿Cómo ver el historial de Git?**

Utiliza el comando:

```
$ git log
```

Este comando muestra una lista detallada de los commits realizados, con sus respectivos identificadores, mensajes y fechas.

- **¿Cómo buscar en el historial de Git?**

Puedes buscar en el historial utilizando el comando:

```
$ git log --grep="término de búsqueda"
```

Este comando filtra los commits que contienen el término especificado en su mensaje.

- **¿Cómo borrar el historial de Git?**

Borrar o reescribir el historial es una operación delicada y, en general, se evita en repositorios colaborativos. Sin embargo, se pueden utilizar comandos como:

```
#Modificar commits recientes
```

```
$ git rebase -i
```

```
#Para reescribir la historia completa
```

```
$ git filter-branch
```

- **¿Qué es un repositorio privado en GitHub?**

Un repositorio privado en GitHub es aquel cuyo contenido solo es accesible a los colaboradores autorizados. Esto permite mantener el código confidencial y restringir el acceso público

- **¿Cómo crear un repositorio privado en GitHub?**

Al crear un nuevo repositorio, selecciona la opción "Private" en la sección de visibilidad. Esto hará que el repositorio sea accesible únicamente para ti y para quienes autorices.

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

Accede al repositorio y haz clic en la pestaña "Settings" (Configuración) -> Selecciona "Manage access" (Administrar acceso) > Haz clic en "Invite a collaborator" (Invitar a un colaborador) -> Ingresa el nombre de usuario o email de la persona a invitar -> Envía la invitación y la persona deberá aceptarla para obtener acceso.

- **¿Qué es un repositorio público en GitHub?**

Un repositorio público es aquel en el que el código es visible para cualquier persona en Internet. Es ideal para proyectos de código abierto donde se busca colaboración y transparencia

- **¿Cómo crear un repositorio público en GitHub?**

Durante el proceso de creación de un repositorio, selecciona la opción "Public" en la sección de visibilidad. Esto hará que el repositorio sea accesible para cualquier usuario.

- **¿Cómo compartir un repositorio público en GitHub?**

Para compartir un repositorio público, basta con copiar la URL del repositorio y enviarla a quien desees. Se puede obtener en la sección que dice "< > Code "

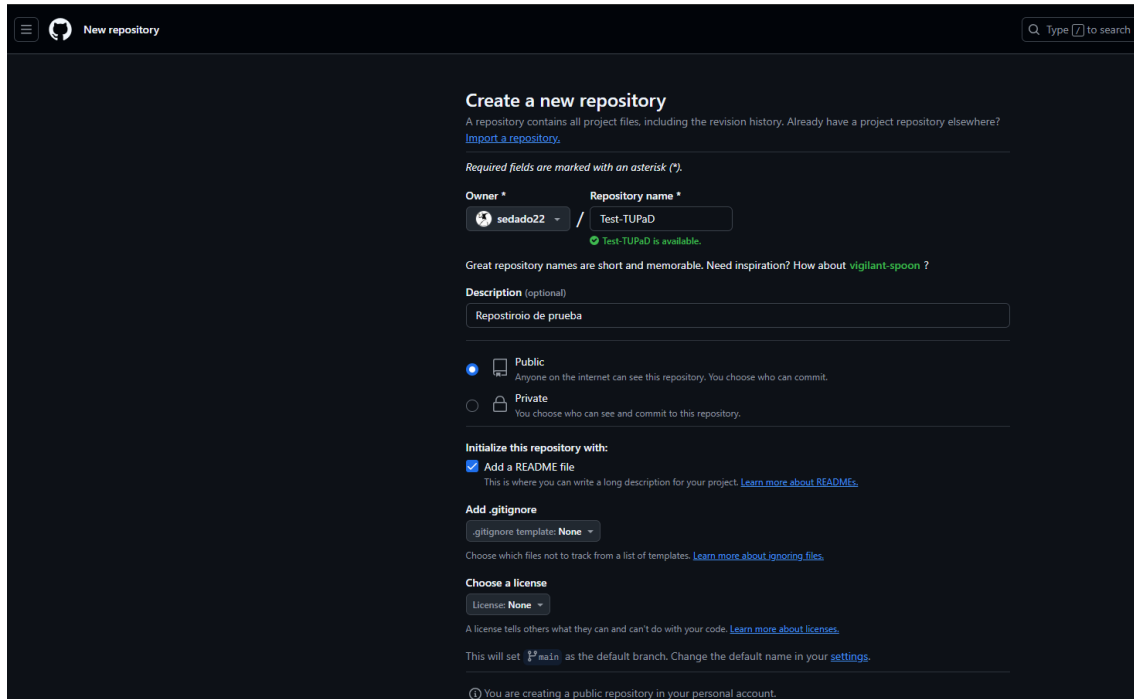
2) Realizar la siguiente actividad:

*Crear un repositorio.

Dale un nombre al repositorio.

Elije el repositorio sea público.

Inicializa el repositorio con un archivo.



The screenshot shows the GitHub 'Create a new repository' interface. At the top, it says 'Create a new repository' and explains that a repository contains all project files, including the revision history. Below this, there are fields for 'Owner' (set to 'sedado22') and 'Repository name' (set to 'Test-TUPaD'). A green checkmark indicates that 'Test-TUPaD is available'. There is a 'Description' field with the text 'Repositorio de prueba'. Under 'Initialize this repository with:', the 'Add a README file' option is selected. There is also a section for 'Add .gitignore' with a dropdown menu set to 'None'. At the bottom, there is a 'Choose a license' section with a dropdown menu set to 'None'. A note at the bottom states 'You are creating a public repository in your personal account.'

*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).

```
CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas
$ pwd
/c/Users/CTI24237/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas
$ git clone https://github.com/sedado22/Test-TUPaD.git
Cloning into 'Test-TUPaD'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas
$ dir
Test-TUPaD

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas
$ cd Test-TUPaD/

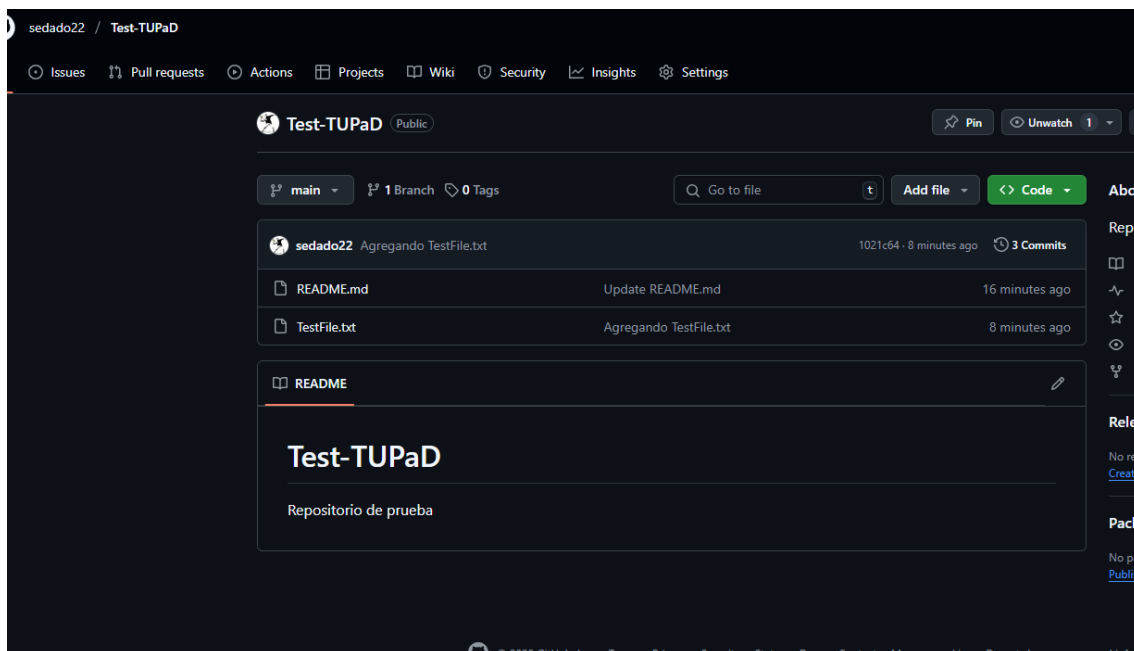
CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (main)
$ echo "Archivo de Prueba" > TestFile.txt

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (main)
$ git add .
warning: in the working copy of 'TestFile.txt', LF will be replaced by CRLF the next time Git touches it

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (main)
$ git commit -m "Agregando TestFile.txt"
[main 1021c64] Agregando TestFile.txt
1 file changed, 1 insertion(+)
create mode 100644 TestFile.txt

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (main)
$ git push origin main
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), 309 bytes | 154.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/sedado22/Test-TUPaD.git
8c968d8..1021c64 main -> main

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (main)
$
```



*Creando Branchs

Crea una branch

Realizar cambios o agregar un archivo

Subi la Branch

```
CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (main)
$ git branch
* main

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (main)
$ git checkout -b newBranch
Switched to a new branch 'newBranch'

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (newBranch)
$ git branch
main
* newBranch

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (newBranch)
$ echo "modifico el archivo desde newBranch" >
.git/      README.md      TestFile.txt

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (newBranch)
$ echo "modifico el archivo desde newBranch" > TestFile.txt

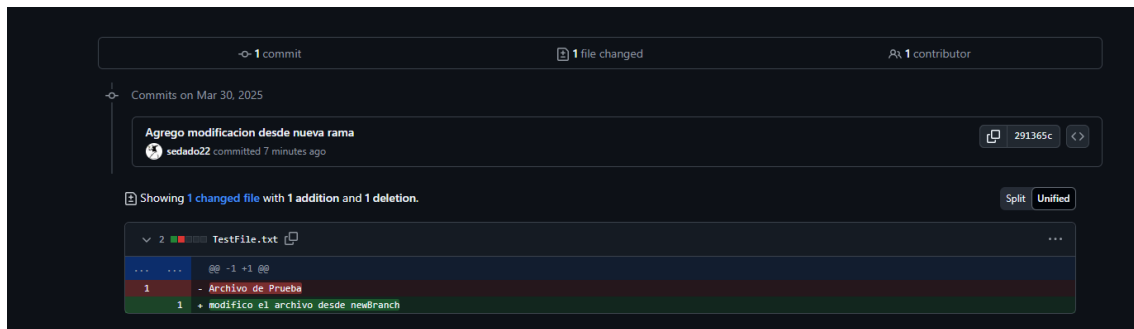
CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (newBranch)
$ git add .
warning: in the working copy of 'TestFile.txt', LF will be replaced by CRLF the next time Git touches it

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (newBranch)
$ git status
On branch newBranch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   TestFile.txt

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (newBranch)
$ git commit -m "Agrego modificacion desde nueva rama"
[newBranch 291365c] Agrego modificacion desde nueva rama
1 file changed, 1 insertion(+), 1 deletion(-)

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/Test-TUPaD (newBranch)
$ git push origin newBranch
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 332 bytes | 332.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'newBranch' on GitHub by visiting:
remote:   https://github.com/sedado22/Test-TUPaD/pull/new/newBranch
remote:
To https://github.com/sedado22/Test-TUPaD.git
 * [new branch]      newBranch -> newBranch
```

The screenshot shows the GitHub web interface for the repository 'Test-TUPaD' owned by 'sedado22'. The repository is public. At the top, there's a navigation bar with links to Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below this, the repository name 'Test-TUPaD' is displayed with a 'Public' badge. A notification bar indicates that 'newBranch' had recent pushes 1 second ago, with a 'Compare & pull request' button. The main content area shows the 'main' branch selected, with 1 branch and 0 tags. A list of recent commits is shown, with the most recent commit by 'sedado22' titled 'Agregando TestFile.txt' from 1021c64, 30 minutes ago, containing 3 commits. Below the commit list, the 'README' file is displayed, showing the title 'Test-TUPaD' and the description 'Repositorio de prueba'.





URL de la actividad:










<https://github.com/sedado22/Test-TUPaD.git>

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".

 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 *

 sedado22 ▾

Repository name *


conflict-exercise


✔ conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about **cautious-memory** ?

Description (optional)

conflict-exercise

☒  **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

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`

```
CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuat
rimestre/PROG1/Pruebas
$ git clone https://github.com/sedado22/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.

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas
$ dir
Test-TUPaD  conflict-exercise

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas
$ 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"`

```
CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/conflict-exercise (feature-branch)
$ dir
README.md

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/conflict-exercise (feature-branch)
$ echo "Esta es una nueva línea" >> README.md

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/conflict-exercise (feature-branch)
$ git add .
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch bf07b1e] 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"`

```
CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/Facu/1er cuatrimestre/PROG1/Pruebas/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/Facu/1er cuatrimestre/PROG1/Pruebas/conflict-exercise (main)
$ echo "Esta es otra línea nueva " >> README.md

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/Facu/1er cuatrimestre/PROG1/Pruebas/conflict-exercise (main)
$ git add .
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/Facu/1er cuatrimestre/PROG1/Pruebas/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 2085ca7] 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.

```
CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/Facu/1er cuatrimestre/PROG1/Pruebas/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.

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/Facu/1er cuatrimestre/PROG1/Pruebas/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"`

```
# conflict-exercise
conflict-exercise
<<<<<< HEAD
Esta es otra linea nueva
=====
Esta es una nueva linea
>>>>>> feature-branch
```

```
# conflict-exercise
conflict-exercise
Esta es una nueva linea
Esta es otra linea nueva
```

```
CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/conflict-exercise (main|MERGING)
$ git add .

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main 0c21b30] Resolved merge conflict

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/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`

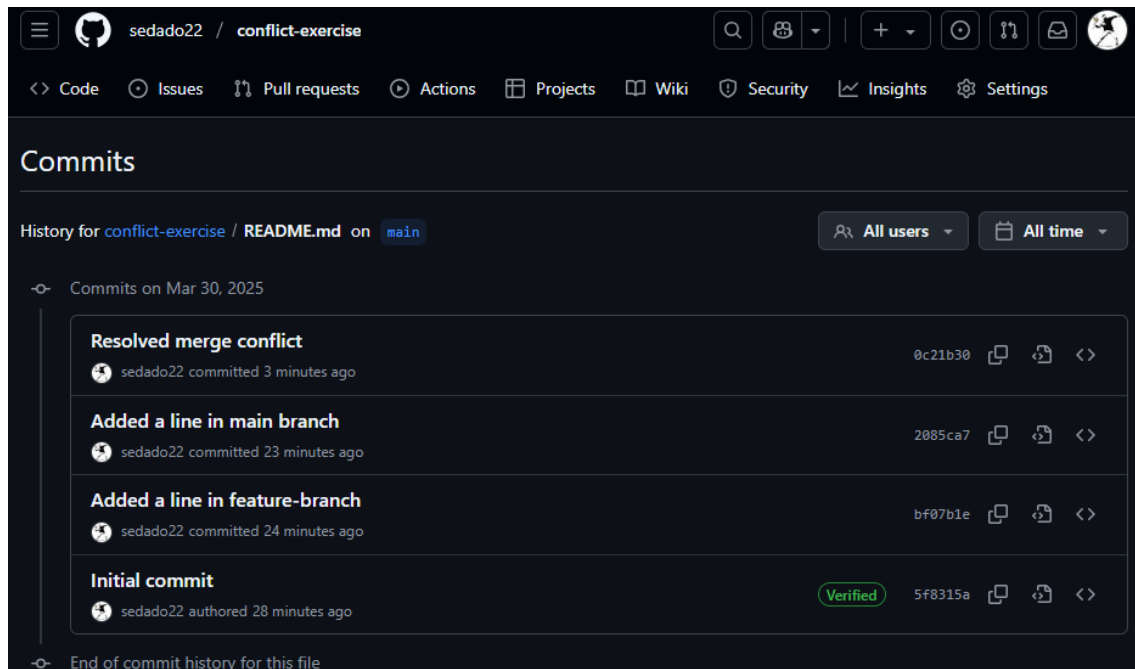
```
CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/conflict-exercise (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 755 bytes | 188.00 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/sedado22/conflict-exercise.git
  5f8315a..0c21b30  main -> main

CTI24237@GBMMHR3 MINGW64 ~/OneDrive - AMX Argentina S.A/Escritorio/facu/1er cuatrimestre/PROG1/Pruebas/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/sedado22/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/sedado22/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.



The screenshot shows the GitHub interface for the repository 'sedado22 / conflict-exercise'. The 'Commits' tab is selected, displaying the commit history for the file 'README.md' on the 'main' branch. The history shows four commits, all by user 'sedado22'. The most recent commit is 'Resolved merge conflict' (hash 0c21b30), followed by 'Added a line in main branch' (2085ca7), 'Added a line in feature-branch' (bf07b1e), and 'Initial commit' (5f8315a, marked as 'Verified'). The interface includes navigation links like Code, Issues, Pull requests, and Actions at the top.

sedado22 / conflict-exercise

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Commits

History for conflict-exercise / README.md on main

All users All time

Commits on Mar 30, 2025

Resolved merge conflict sedado22 committed 3 minutes ago	0c21b30	Copy Diff Code
Added a line in main branch sedado22 committed 23 minutes ago	2085ca7	Copy Diff Code
Added a line in feature-branch sedado22 committed 24 minutes ago	bf07b1e	Copy Diff Code
Initial commit sedado22 authored 28 minutes ago	5f8315a Verified	Copy Diff Code

End of commit history for this file

URL de la actividad :

<https://github.com/sedado22/conflict-exercise.git>