ISFDyT N°70

Tecnicatura Superior en Análisis de Sistemas

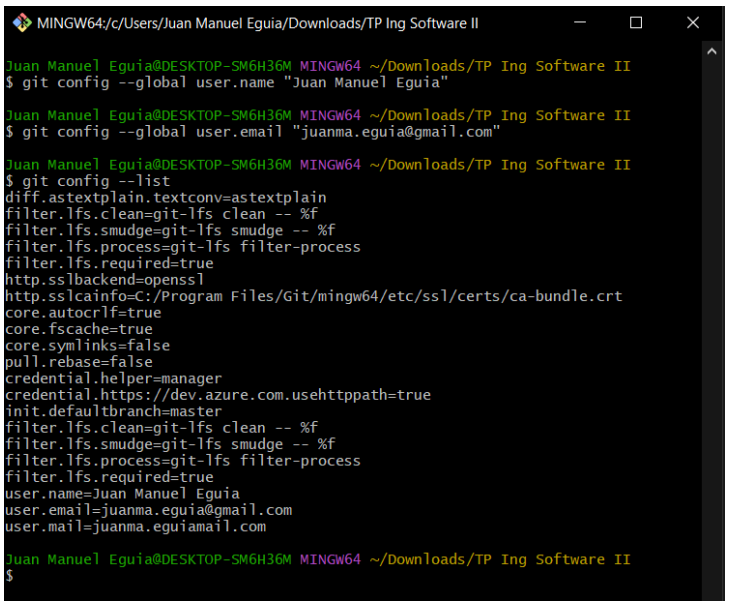
Ingeniería de Software II

- Trabajo Práctico 1 – Control de versiones con Git y GitHub
- Eguia Juan Manuel; Falabella Valentino
- 3° año
- **Nombre de la profesora:** Marina Caseres
- **Fecha de entrega:** 11/07/2025
- Enlace al repositorio GitHub del grupo:
- juanma.eguia@gmail.com ; valentinofala@gmail.com

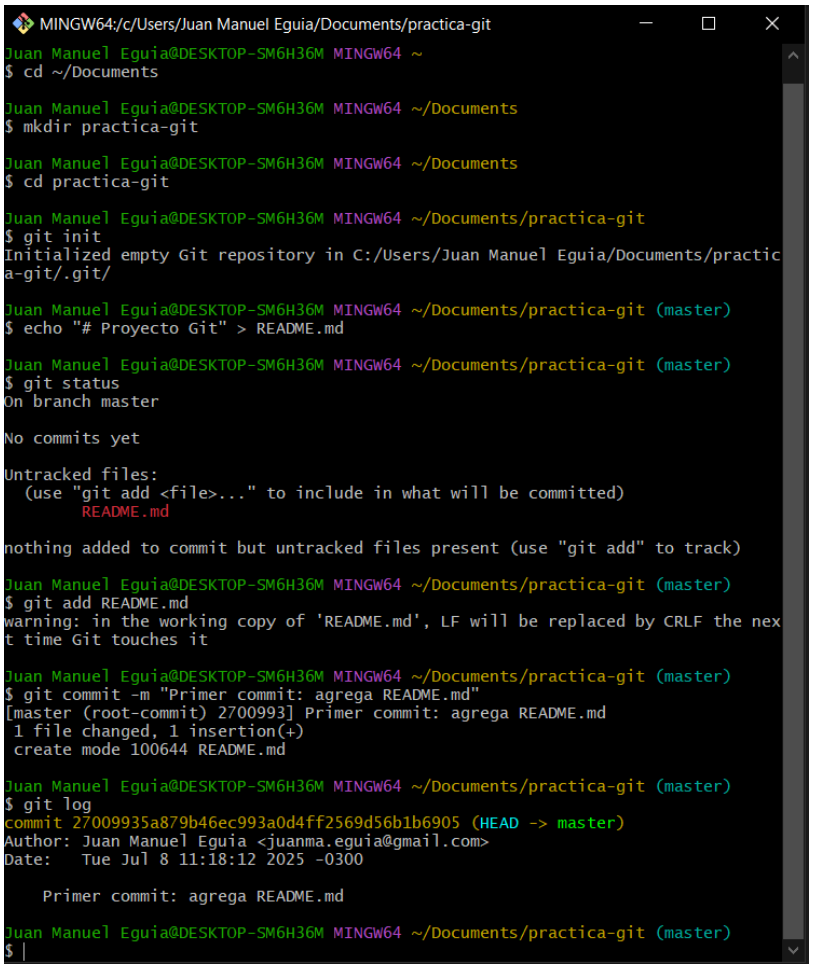
Trabajo Practico 1 – Control de versiones con Git y GitHub

Parte 1: Iniciación a Git y configuración básica

Consigna 1: Configurar Git en tu equipo



Consigna 2: Crear un repositorio local y realizar tu primer commit



¿Dónde queda guardado el control de versiones? ¿En la nube o en tu equipo?

El control de versiones queda guardado en el equipo, dentro de la carpeta del repositorio local.

¿Qué información te brinda git status? ¿Por qué es útil antes de hacer un commit?

Muestra el estado actual de los archivos en el repositorio. Es útil antes de hacer un commit porque permite revisar que cambios se registrarán.

¿Cuál es la diferencia entre agregar (add) y guardar un cambio (commit)? ¿Podés hacer un commit sin hacer un add antes?

Git add prepara los archivos para ser incluidos en el próximo commit, mientras que commit guarda esos cambios en el historial. No se puede hacer un commit de cambios nuevos si no se han agregado previamente con add.

¿Qué información muestra este comando? (git log)

Muestra el historial de commits del repositorio. Sirve para rastrear que cambios se hicieron y cuando.

Parte 2: Repositorio remoto y sincronización

Consigna 3: Crear un repositorio remoto y vincularlo

```
Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (master)
$ git remote add origin https://github.com/juaneguia/practica-git.git

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (master)
$ git branch -M main

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (main)
$ git push -u origin main
remote: Repository not found.
fatal: repository 'https://github.com/juaneguia/practica-git.git/' not found

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (main)
$ git remote add origin https://github.com/juaneguia/practica-git.git
error: remote origin already exists.

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (main)
$ git branch -M main

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 249 bytes | 249.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/juaneguia/practica-git.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (main)
$ git pull origin main
From https://github.com/juaneguia/practica-git
 * branch            main       -> FETCH_HEAD
Already up to date.
```

**Inicialmente ejecuté los comandos antes de haber creado el repositorio en GitHub.*

¿Qué significa vincular tu repositorio local con uno remoto? ¿Podés trabajar sin conexión a internet después de hacer esto?

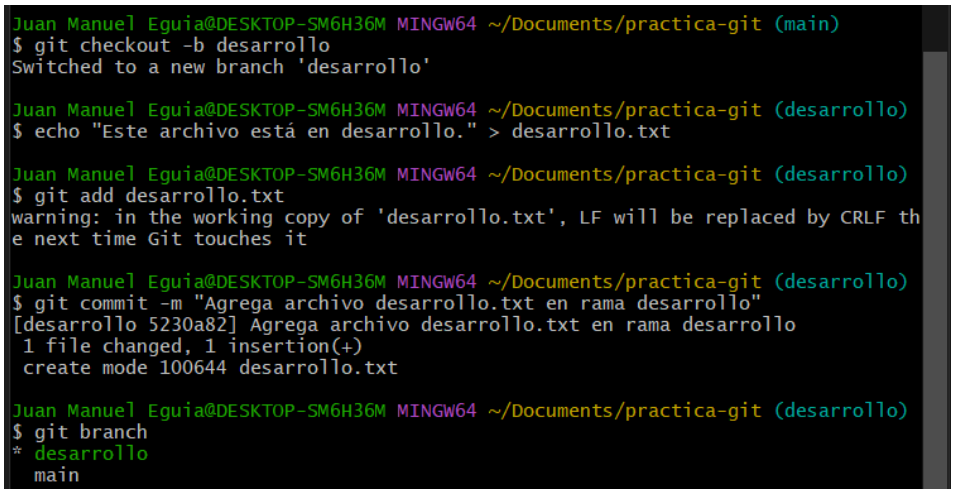
Vincular el repositorio local con uno remoto significa conectar tu proyecto en tu computadora con una copia alojada en GitHub. Esto permite subir (push) o traer (pull) cambios. Sí, se puede seguir trabajando sin conexión: los cambios se guardan localmente y se sincronizan cuando haya internet.

¿Qué estás logrando al hacer un push? ¿Por qué hay que subir los cambios al repositorio remoto si ya están en tu computadora?

Con git push estás enviando tus commits locales al repositorio remoto en GitHub. Subir los cambios es importante para tener una copia de respaldo en la nube, colaborar con otros y mantener actualizados todos los entornos que usan ese repositorio.

Parte 3: Trabajo con ramas

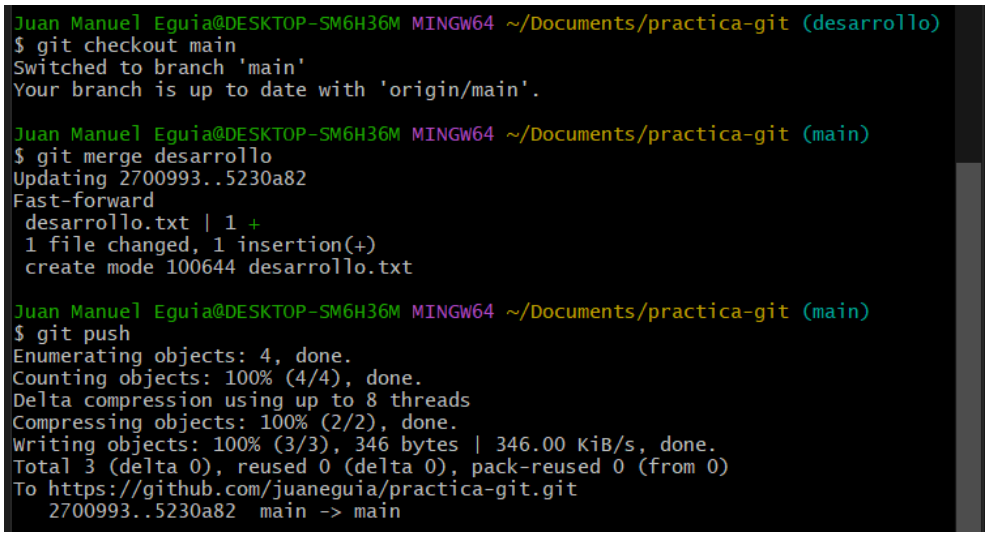
Consigna 4: Crear y trabajar en una nueva rama



¿Para qué sirve crear una nueva rama? ¿Qué ventajas tiene trabajar en ramas distintas en lugar de hacerlo siempre en la principal?

Crear una nueva rama permite desarrollar funcionalidades o hacer pruebas sin afectar la rama principal. Trabajar en ramas distintas evita errores en el código principal, facilita la colaboración y hace más fácil revisar, probar y fusionar cambios.

Consigna 5: Fusionar ramas



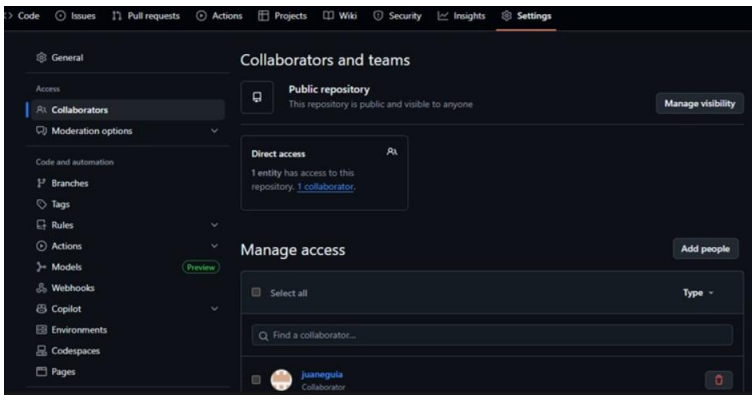
¿Qué pasa si hay diferencias entre las ramas al hacer un merge? ¿Podés revertir un merge si algo sale mal?)

Si hay diferencias en los mismos archivos o líneas, Git genera un conflicto que debe resolverse manualmente antes de completar el merge. Si algo sale mal, es posible revertir el merge usando git reset o git revert, aunque conviene tener un commit de respaldo antes de fusionar.

Parte 4: Colaboración y trabajo en equipo

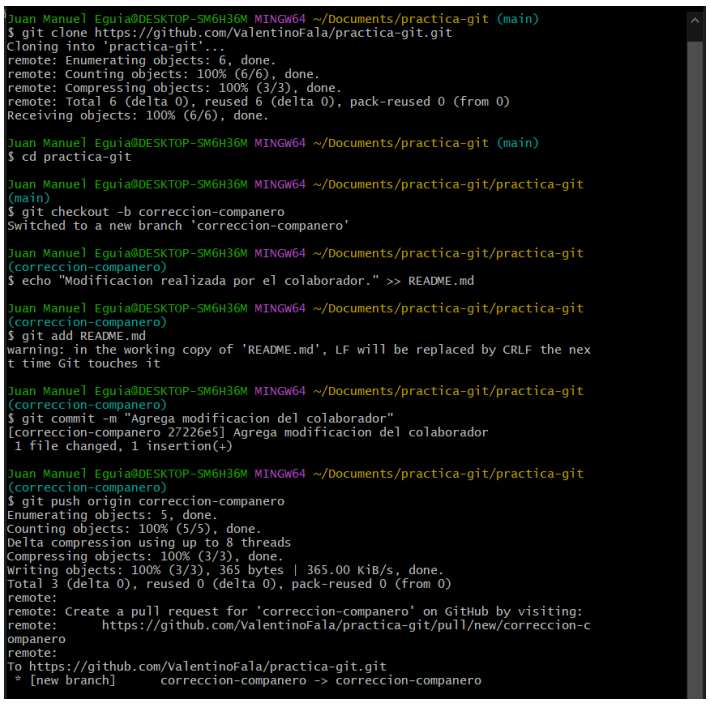
Consigna 6: Agregar un colaborador

En este punto, mi compañero me agrego como colaborador y yo voy a clonar su repositorio.



Mi repositorio (practica-git) > Settings > Collaborators > Add people > nombre del colaborador (juaneguia)

Consigna 7: Clonar un repositorio como colaborador



¿En qué se diferencia clonar un repositorio de crear uno nuevo? ¿Por qué es importante clonar en lugar de descargar un ZIP?

Clonar crea una copia exacta del repositorio remoto incluyendo todo el historial de commits, ramas y configuración de Git. En cambio, descargar un ZIP solo baja los archivos sin historial ni conexión al repositorio. Clonar es esencial para contribuir y sincronizar cambios correctamente.

Consigna 8: Pull Request y revisión de cambios

El propietario del GitHub hizo el Pull Request, se aprobó el cambio y se fusiono el Pull.

¿Qué ventajas tiene hacer una revisión del código antes de fusionarlo? ¿Qué actitudes o prácticas colaborativas se ponen en juego al hacer un Pull Request?

Revisar el código antes de fusionarlo permite detectar errores, mejorar la calidad del código y asegurar que los cambios no rompan el proyecto. Fomenta la comunicación, el respeto por el trabajo en equipo, la responsabilidad compartida y el aprendizaje entre pares.

SUBIMOS TRABAJOS (ACT 1, 2 y 3) como ramas

```
(correccion-companero)
$ git checkout -b juan
Switched to a new branch 'juan'

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git/practica-git
(juan)
$ git add Informe_Git_Eguia_Falabella
fatal: pathspec 'Informe_Git_Eguia_Falabella' did not match any files

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git/practica-git
(juan)
$ git add Informe_Git_Eguia_Falabella.pdf

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git/practica-git
(juan)
$ git commit -m "Agrego pdf Eguia"
[juan c8fff07] Agrego pdf Eguia
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Informe_Git_Eguia_Falabella.pdf

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git/practica-git
(juan)
$ git branch
  correccion-companero
* juan
  main

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git/practica-git
(juan)
$ git push
fatal: The current branch juan has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin juan

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.


Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git/practica-git
(juan)
$ git push origin juan
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 330.28 KiB | 30.02 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'juan' on GitHub by visiting:
remote:      https://github.com/ValentinoFala/practica-git/pull/new/juan
remote:
To https://github.com/ValentinoFala/practica-git.git
 * [new branch]      juan -> juan
```

Luego de hacer las ramas y subir los PDF, hicimos el Pull Request y fusionamos loas ramas.

Parte 5: Conflictos y resolución

Consigna 9: Generar y resolver conflictos de merge

Mi compañero creo el conflicto.txt en la main

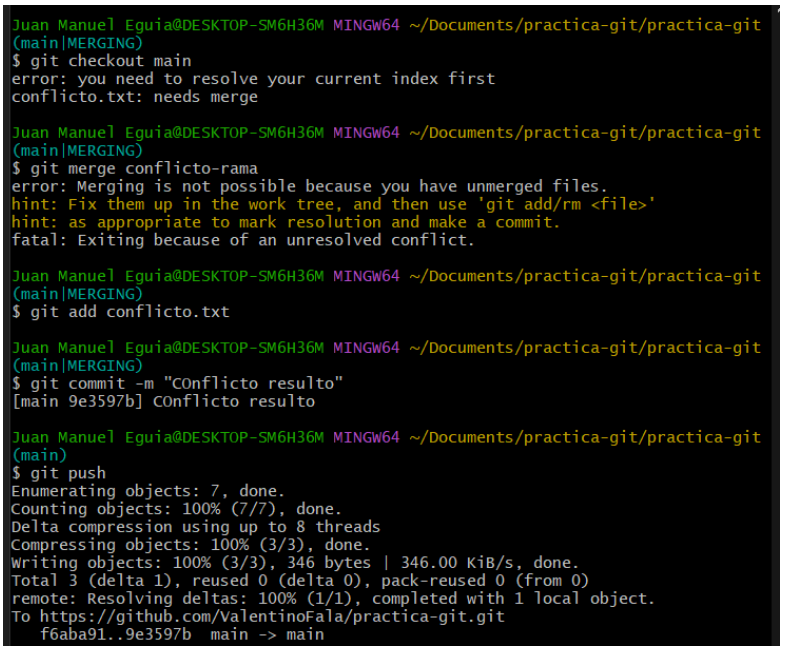
```
Educacion@DESKTOP-PPICMMB MINGW64 ~/Documents/practica-git (main)
$ git checkout main
D       conflicto.txt
Already on 'main'
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

Educacion@DESKTOP-PPICMMB MINGW64 ~/Documents/practica-git (main)
$ echo "Linea modificada desde main" > conflicto.txt

Educacion@DESKTOP-PPICMMB MINGW64 ~/Documents/practica-git (main)
$ git add conflicto.txt
warning: in the working copy of 'conflicto.txt', LF will be replaced by CRLF the
 next time Git touches it

Educacion@DESKTOP-PPICMMB MINGW64 ~/Documents/practica-git (main)
$ git commit -m "Cambio en main que causará conflicto"
[main f6aba91] Cambio en main que causará conflicto
 1 file changed, 1 insertion(+), 1 deletion(-)

Educacion@DESKTOP-PPICMMB MINGW64 ~/Documents/practica-git (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% (7/7), done.
Writing objects: 100% (9/9), 1.05 KiB | 178.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/ValentinoFala/practica-git.git
   5503035..f6aba91  main -> main
```

Cree una rama (conflicto-rama)

```
Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git/practica-git
(juan)
$ git checkout -b conflicto-rama
Switched to a new branch 'conflicto-rama'

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git/practica-git
(conflicto-rama)
$ echo "Linea diferente modificada en rama" > conflicto.txt

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git/practica-git
(conflicto-rama)
$ git add conflicto.txt
warning: in the working copy of 'conflicto.txt', LF will be replaced by CRLF the
 next time Git touches it

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git/practica-git
(conflicto-rama)
$ git commit -m "Cambio en rama que causara conflicto"
[conflicto-rama 7f9f079] Cambio en rama que causara conflicto
 1 file changed, 1 insertion(+)
 create mode 100644 conflicto.txt
```

```
Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git/practica-git
(conflicto-rama)
$ git push origin conflicto-rama
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 327 bytes | 327.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'conflicto-rama' on GitHub by visiting:
remote:      https://github.com/ValentinoFala/practica-git/pull/new/conflicto-ra
ma
remote:
To https://github.com/ValentinoFala/practica-git.git
 * [new branch]      conflicto-rama -> conflicto-rama
```

Cuando nos dio error, modificamos el txt para que nos permita ejecurtar los cambios.



¿Qué ocurre si otra persona hizo cambios en GitHub y vos no hiciste un pull antes de seguir trabajando?

Si otra persona sube cambios y vos no hacés un git pull, es posible que tu repositorio quede desactualizado. Esto puede provocar conflictos cuando intentes subir tus cambios, ya que Git detecta que hay diferencias entre tu rama local y la remota.

¿Qué riesgos hay si no sincronizás antes de hacer cambios?

El principal riesgo es sobrescribir el trabajo de otro sin darte cuenta. También podés tener errores al hacer push, conflictos difíciles de resolver o perder tiempo arreglando problemas que se evitan fácilmente sincronizando antes (git pull). Es una buena práctica **actualizar siempre antes de modificar**.

Reflexión final sobre lo aprendido

A lo largo de este trabajo aprendimos a configurar Git en nuestros equipos, crear repositorios locales y remotos, y realizar commits para registrar cambios de forma ordenada. Uno de los aspectos más importantes fue entender la diferencia entre trabajar en una sola rama y utilizar ramas separadas, lo cual permite trabajar de forma más segura y organizada. También resultó muy útil aprender a vincular el repositorio local con uno remoto, algo indispensable para colaborar en proyectos reales.

Durante el trabajo en equipo, pude experimentar prácticas importantes en entornos colaborativos. Al final, enfrentar un conflicto de merge y resolverlo manualmente fue una de las instancias más complejas del trabajo, pero esto nos permitió ver cómo Git protege el historial del proyecto y obliga a resolver los problemas de forma consciente y controlada.