

# ACTIVIDAD 5

Integrantes:

- Diego Manuel Delgado Velarde

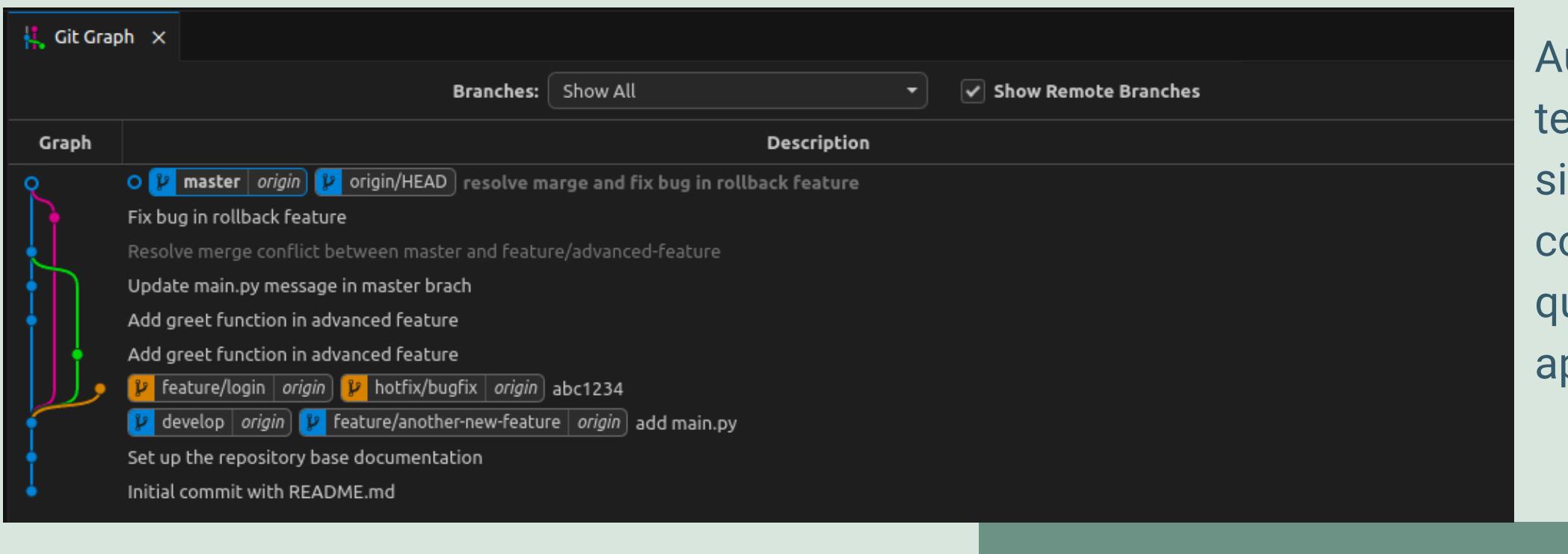
EJERCICIO: CLONA UN  
REPOSITORIO GIT CON  
MÚLTIPLES RAMAS.

## 1. Clonar un repositorio con múltiples ramas.

```
diegodev@HPavilion:~/Desktop/dev-practice$ git clone git@github.com:GuidoCh23/PruebasDesarrolloDeSoftware.git
Cloning into 'PruebasDesarrolloDeSoftware'...
remote: Enumerating objects: 31, done.
remote: Counting objects: 100% (31/31), done.
remote: Compressing objects: 100% (13/13), done.
Receiving objects: 100% (31/31), done.
Resolving deltas: 100% (13/13), done.
remote: Total 31 (delta 13), reused 29 (delta 11), pack-reused 0 (from 0)
```

```
diegodev@HPavilion:~/Desktop/dev-practice/PruebasDesarrolloDeSoftware$ git branch
  develop
  feature/another-new-feature
  feature/login
  hotfix/bugfix
* master
```

## 2. Identifica dos ramas que puedes fusionar utilizando git merge --ff.

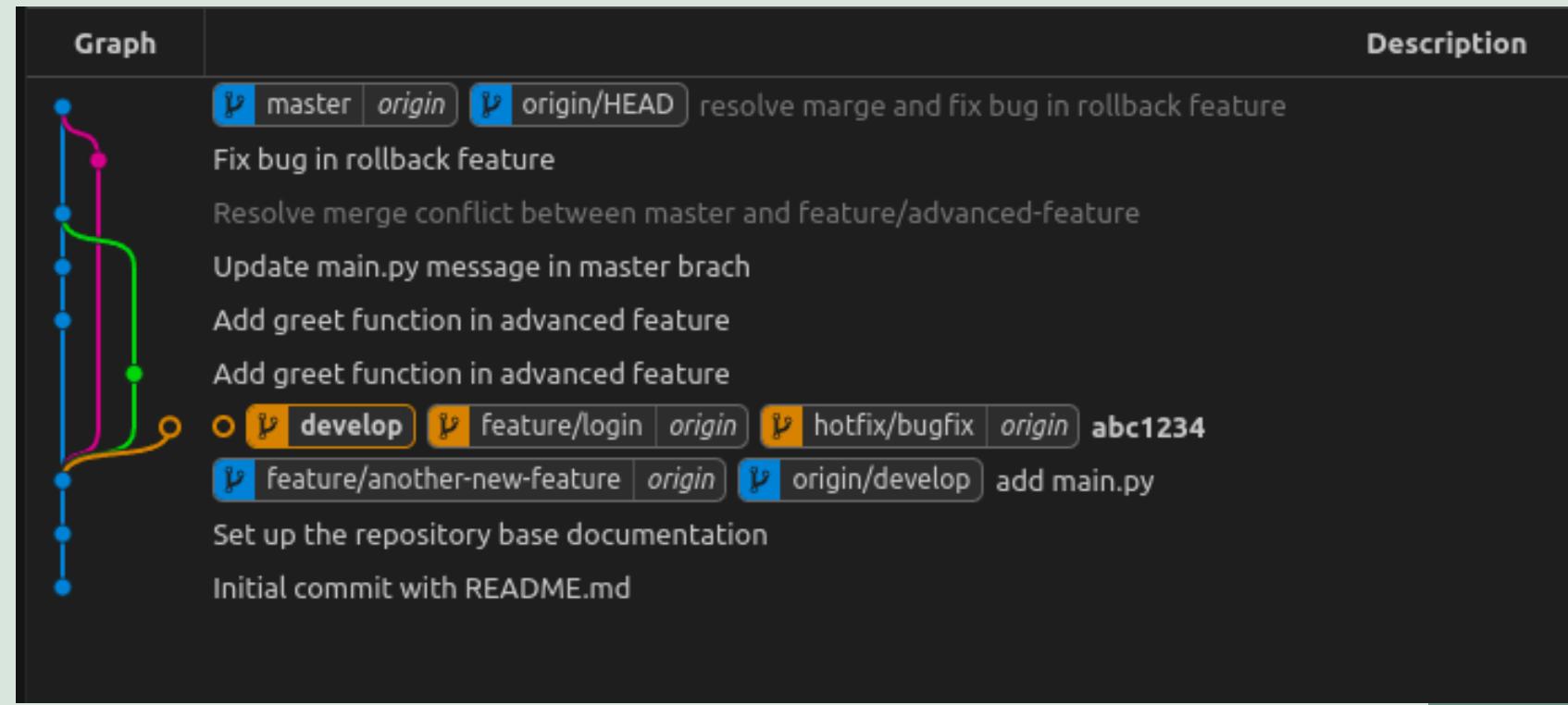


Aunque no es recomendable usar merge --ff cuando tenemos multiples ramas que tienen cambios simultaneos, en este caso identificamos una rama, como **develop**, que no recibio ningun commit desde que se creo la rama **feature/login**, entonces podemos aprovechar ello y hacer un git merge --ff

### 3. Haz el proceso de fusión utilizando git merge --ff.

```
diegodev@HPavilion:~/Desktop/dev-practice/PruebasDesarrolloDeSoftware$ git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.
diegodev@HPavilion:~/Desktop/dev-practice/PruebasDesarrolloDeSoftware$ git merge --ff feature/login
Updating 19474a5..4c35195
Fast-forward
 ABC1234.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 ABC1234.md
```

### 4. Verifica el historial con git log --graph --oneline.



```
diegodev@HPavilion:~/Desktop/dev-practice/PruebasDesarrolloDeSoftware$ git log --graph --oneline --all
* af57515 (origin/master, origin/HEAD, master) resolve marge and fix bug in rollback feature
|\ \
| * 1160b06 Fix bug in rollback feature
| * fe34bde Resolve merge conflict between master and feature/advanced-feature
| |\ \
| | * 4f4a679 Add greet function in advanced feature
| | |
| | * 01d7d20 Update main.py message in master brach
| | fe87c76 Add greet function in advanced feature
| |
| * 4c35195 (HEAD -> develop, origin/hotfix/bugfix, origin/feature/login, hotfix/bugfix, feature/login) abc1234
|/
* 19474a5 (origin/feature/another-new-feature, origin/develop, feature/another-new-feature) add main.py
* 7c10371 Set up the repository base documentation
* cf05b8d Initial commit with README.md
diegodev@HPavilion:~/Desktop/dev-practice/PruebasDesarrolloDeSoftware$ git log --graph --oneline
* 4c35195 (HEAD -> develop, origin/hotfix/bugfix, origin/feature/login, hotfix/bugfix, feature/login) abc1234
* 19474a5 (origin/feature/another-new-feature, origin/develop, feature/another-new-feature) add main.py
* 7c10371 Set up the repository base documentation
* cf05b8d Initial commit with README.md
```

# PREGUNTA

¿En qué situaciones recomendarías evitar el uso de git merge --ff? Reflexiona sobre las desventajas de este método.

Cuando estamos trabajando en proyectos colaborativos que tienen multiples cambios en multiples ramas, aparte que no da el commit extra que tiene el no-fast-forward para saber el punto de fusion que ayuda en trabajos colaborativos. En conclusion, no es optimo para proyectos colaborativos complejos.

EJERCICIO: SIMULA UN FLUJO  
DE TRABAJO DE EQUIPO.

1. Trabaja en dos ramas independientes, creando diferentes cambios en cada una.

```
diegodev@HPavilion:~/Desktop/dev-practice$ mkdir prueba-no-fast-forward-merge
diegodev@HPavilion:~/Desktop/dev-practice$ cd prueba-no-fast-forward-merge
diegodev@HPavilion:~/Desktop/dev-practice/prueba-no-fast-forward-merge$ git init
Initialized empty Git repository in /home/diegodev/Desktop/dev-practice/prueba-no-fast-forward-merge/.git/
diegodev@HPavilion:~/Desktop/dev-practice/prueba-no-fast-forward-merge$ echo "# Mi Proyecto" > README.md
diegodev@HPavilion:~/Desktop/dev-practice/prueba-no-fast-forward-merge$ git add README.md
diegodev@HPavilion:~/Desktop/dev-practice/prueba-no-fast-forward-merge$ git commit -m "Commit inicial en main"
[main (root-commit) 70823ea] Commit inicial en main
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
diegodev@HPavilion:~/Desktop/dev-practice/prueba-no-fast-forward-merge$ git checkout -b add-feature
Switched to a new branch 'add-feature'
diegodev@HPavilion:~/Desktop/dev-practice/prueba-no-fast-forward-merge$ echo "Implementando una nueva característica..." >> README.md
diegodev@HPavilion:~/Desktop/dev-practice/prueba-no-fast-forward-merge$ git add README.md
diegodev@HPavilion:~/Desktop/dev-practice/prueba-no-fast-forward-merge$ git commit -m "Implementar nueva característica"
[add-feature c3b64dd] Implementar nueva característica
 1 file changed, 1 insertion(+)
diegodev@HPavilion:~/Desktop/dev-practice/prueba-no-fast-forward-merge$ git log --oneline
c3b64dd (HEAD -> add-feature) Implementar nueva característica
70823ea (main) Commit inicial en main
diegodev@HPavilion:~/Desktop/dev-practice/prueba-no-fast-forward-merge$
```

2. Fusiona ambas ramas con git merge --no-ff para ver cómo se crean los commits de fusión.

```
diegodev@HPavillon:~/Desktop/dev-practice/prueba-no-fast-forward-merge$ git checkout main
Switched to branch 'main'
diegodev@HPavillon:~/Desktop/dev-practice/prueba-no-fast-forward-merge$ git merge --no-ff add-feature
Merge made by the 'ort' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
```

3. Observa el historial utilizando git log --graph --oneline.

```
diegodev@HPavillon:~/Desktop/dev-practice/prueba-no-fast-forward-merge$ git log --graph --oneline
*   d414155 (HEAD -> main) Merge branch 'add-feature'
|\ 
| * c3b64dd (add-feature) Implementar nueva característica
|/
* 70823ea Commit inicial en main
```

# PREGUNTAS

¿Cuáles son las principales ventajas de utilizar git merge -no-ff en un proyecto en equipo?

Nos permite tener ese commit extra al fusionar ramas permitiendo que el equipo entienda perfectamente el punto de integracion, sabiendo que ramas se fusionaron , cuando y porque, brindando una mejor trazabilidad.

¿Qué problemas podrían surgir al depender excesivamente de commits de fusión?

Los problemas seria que se desordene el historial debido a demasiados commits de fusion dificultando la lectura, por ello los equipos deben balancear el uso de no-fast-forward, netamente para fusiones clave, con la necesidad de mantener un historial manejable.

EJERCICIO: CREA MÚLTIPLES  
COMMITs EN UNA RAMA.

## 1. Haz varios cambios y commits en una rama feature.

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git add README.md
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git commit -m "Commit inicial en main"
[main (root-commit) 69257f8] Commit inicial en main
 1 file changed, 1 insertion(+)
  create mode 100644 README.md
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git checkout -b add-basic-files
Switched to a new branch 'add-basic-files'
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ echo "# CÓMO CONTRIBUIR" >> CONTRIBUTING.md
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git add CONTRIBUTING.md
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git commit -m "Agregar CONTRIBUTING.md"
[add-basic-files b8ed470] Agregar CONTRIBUTING.md
 1 file changed, 1 insertion(+)
  create mode 100644 CONTRIBUTING.md
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ echo "# LICENCIA" >> LICENSE.txt
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git add LICENSE.txt
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git commit -m "Agregar LICENSE.txt"
[add-basic-files 64bbcb3] Agregar LICENSE.txt
 1 file changed, 1 insertion(+)
  create mode 100644 LICENSE.txt
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git log --oneline
64bbcb3 (HEAD -> add-basic-files) Agregar LICENSE.txt
b8ed470 Agregar CONTRIBUTING.md
69257f8 (main) Commit inicial en main
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ █
```

## 2. Fusiona la rama con git merge --squash para aplanar todos los commits en uno solo.

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git merge --squash add-basic-files
Updating 69257f8..64bbcb3
Fast-forward
Squash commit -- not updating HEAD
CONTRIBUTING.md | 1 +
LICENSE.txt     | 1 +
2 files changed, 2 insertions(+)
create mode 100644 CONTRIBUTING.md
create mode 100644 LICENSE.txt
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git status
On branch main
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
  new file:   CONTRIBUTING.md
  new file:   LICENSE.txt

diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git add .
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git commit -m "Agregar documentación estándar del repositorio"
[main dd6f8be] Agregar documentación estándar del repositorio
 2 files changed, 2 insertions(+)
  create mode 100644 CONTRIBUTING.md
  create mode 100644 LICENSE.txt
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ █
```

## 3. Verifica el historial de commits antes y después de la fusión para ver la diferencia.

```
* main
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git checkout add-basic-files
Switched to branch 'add-basic-files'
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git log --oneline
64bbcb3 (HEAD -> add-basic-files) Agregar LICENSE.txt
b8ed470 Agregar CONTRIBUTING.md
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git branch
  add-basic-files
* main
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git checkout add-basic-files
Switched to branch 'add-basic-files'
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ git log --oneline
64bbcb3 (HEAD -> add-basic-files) Agregar LICENSE.txt
b8ed470 Agregar CONTRIBUTING.md
69257f8 Commit inicial en main
diegodev@HPavilion:~/Desktop/dev-practice/prueba-squash-merge$ █
```

# PREGUNTAS

¿Cuándo es recomendable utilizar una fusión squash?

Es recomendable cuando tienes muchos commits pequeños o experimentales que no merecen estar en el historial principal. En consecuencia mantiene la limpieza del historial al integrar ramas de características.ideal para proyectos donde se prioriza la simplicidad y claridad en el código base.

¿Qué ventajas ofrece para proyectos grandes en comparación con fusiones estándar?

Mantiene el historial principal limpio y navegable ya que evita la acumulacion excesiva de commits pequeños, priorizando la simplicidad y claridad , ya que todo estos commits pequeños lo agrupa en un solo commit, haciendo que la rama principal se mantenga enfocada solo en los cambios finales e importantes.

# EJERCICIO: RESOLVER CONFLICTOS EN UNA FUSIÓN NON-FAST-FORWARD

En algunos casos, las fusiones no son tan sencillas y pueden surgir conflictos que necesitas resolver manualmente.

Este ejercicio te guiará a través del proceso de manejo de conflictos.

1.

Inicia un nuevo repositorio:  
mkdir prueba-merge-conflict  
cd prueba-merge-conflict  
git init

```
diegodev@HPavilion:~/Desktop/dev-practice$ mkdir prueba-merge-conflict
diegodev@HPavilion:~/Desktop/dev-practice$ cd prueba-merge-conflict
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git init
Initialized empty Git repository in /home/diegodev/Desktop/dev-practice/prueba-merge-conflict/.git/
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$
```

2.

Crea un archivo index.html y realiza un commit en la rama main:  
echo "<html><body><h1>Proyecto inicial CC3S2</h1></body></html>" > index.html  
git add index.html  
git commit -m "commit inicial del index.html en main"

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ echo "<html><body><h1>Proyecto inicial CC3S2</h1></body></html>" > index.html
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git add index.html
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git commit -m "commit inicial del index.html en main"
[main (root-commit) 1ad895a] commit inicial del index.html en main
 1 file changed, 1 insertion(+)
   create mode 100644 index.html
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$
```

3.

Crea y cambia a una nueva rama feature-update:

git checkout -b feature-update

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git checkout -b feature-update
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git branch
* feature-update
  main
```

4.

Edita el archivo y realiza un commit en la rama

feature-update:

echo "<p>.....</p>" >> index.html

git add index.html

git commit -m "Actualiza ..."

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ echo "<p>Principal Section</p>" >> index.html
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git add index.html
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git commit -m "add principal section in index file"
[feature-update a21b7ae] add principal section in index file
 1 file changed, 1 insertion(+)
```

5.

Regresa a la rama main y realiza una edición en el mismo archivo:

```
git checkout main  
echo "<footer>Contacta aquí:  
example@example.com</footer>" >> index.html  
git add index.html  
git commit -m "...index.html"
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git checkout main  
Switched to branch 'main'  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ echo "<footer>Contacta aquí: example@example.com</footer>" >> index.html  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git add index.html  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git commit -m "Agrega footer de contacto en index.html"  
[main 2cc335e] Agrega footer de contacto en index.html  
 1 file changed, 1 insertion(+)
```

6.

Fusiona la rama feature-update con --no-ff y observa el conflicto:

```
git merge --no-ff feature-update
```

7.

Git detectará un conflicto en index.html. Abre el archivo y resuelve el conflicto. Elimina las líneas de conflicto generadas por Git (<<<<<, =====, >>>>>) y crea la versión final del archivo con ambos cambios:

```
<html>  
  <body>  
    <h1>....</h1>  
    <p>....</p>  
    <footer>...example@example.com</footer>  
  </body>  
</html>
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git merge --no-ff feature-update  
< index.html > ...  
You, 5 minutes ago | 1 author (You)  
1  <html>  
2    <body>  
3      <h1>Proyecto inicial CC3S2</h1>  
4      <p>Principal Section</p>  
5      <footer>Contacta aquí: example@example.com</footer>  
6    </body>  
7  </html>  
8
```

# 8.

Agrega el archivo corregido y completa la fusión:

```
git add index.html  
git commit
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git add index.html  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git commit -m "merge with feature-update and resolve  
flicts in index file"  
[main 194a215] merge with feature-update and resolve conflicts in index file  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ █
```

# 9.

Verifica el historial para confirmar la fusión y el commit de resolución de conflicto:

```
git log --graph --oneline
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ git log --graph --oneline  
*   194a215 (HEAD -> main) merge with feature-update and resolve conflicts in index file  
|\  
| * a21b7ae (feature-update) add principal section in index file  
* | 2cc335e Agrega footer de contacto en index.html  
|/  
* 1ad895a commit inicial del index.html en main  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-merge-conflict$ █
```

# PREGUNTAS

- ¿Qué pasos adicionales tuviste que tomar para resolver el conflicto?

Abrir el archivo en vscode e identificar los delimitadores para ver como combinar el codigo de ambas ramas de acuerdo al objetivo por el cual realizo la fusion, luego agrego el archivo y hago un commit.

- ¿Qué estrategias podrías emplear para evitar conflictos en futuros desarrollos colaborativos?

Bueno coordinar con mi equipo para que cada uno trabaje en una rama especifica y que avise cuando quiera modificar un archivo que es comun a los demás integrantes del equipo,mediante Pull Requests y poder realizar los merges en la rama principal disminuyendo asi el numero de conflictos.

# EJERCICIO: COMPARAR LOS HISTORIALES CON GIT LOG DESPUÉS DE DIFERENTES FUSIONES

Este ejercicio te permitirá observar las diferencias en el historial generado por fusiones fast-forward, non-fast-forward y squash.

# 1.

Crea un nuevo repositorio y realiza varios commits en dos ramas:

```
$ mkdir prueba-compare-merge  
$ cd prueba-compare-merge  
$ git init  
$ echo "Version 1.0" > version.txt  
$ git add version.txt  
$ git commit -m "...."  
$ git checkout -b feature-1  
$ echo "Caracteristica 1 agregada" >> version.txt  
$ git add version.txt  
$ git commit -m "Aregar caracteristica 1"  
$ git checkout main  
$ git checkout -b feature-2  
$ echo "Caracteristica 2 agregada" >> version.txt  
$ git add version.txt  
$ git commit -m "Se agrega caracteristica 2"
```

```
diegodev@HPavilion:~/Desktop/dev-practice$ mkdir prueba-compare-merge  
diegodev@HPavilion:~/Desktop/dev-practice$ cd prueba-compare-merge  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git init  
Initialized empty Git repository in /home/diegodev/Desktop/dev-practice/prueba-compare-merge/.git/  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ echo "Version 1.0" > version.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git add version.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git commit -m "...."  
[main (root-commit) beaa8f1] ....  
 1 file changed, 1 insertion(+)  
  create mode 100644 version.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git checkout -b feature-1  
Switched to a new branch 'feature-1'  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ echo "Caracteristica 1 agregada" >> version.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git add version.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git commit -m "Aregar caracteristica 1"  
[feature-1 61a7fb0] Agregar caracteristica 1  
 1 file changed, 1 insertion(+)  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git checkout main  
Switched to branch 'main'  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git checkout -b feature-2  
Switched to a new branch 'feature-2'  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ echo "Caracteristica 2 agregada" >> version.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git add version.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git commit -m "Se agrega caracteristica 2"  
[feature-2 db8c928] Se agrega caracteristica 2  
 1 file changed, 1 insertion(+)  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ █
```

2.

Fusiona feature-1 usando fast-forward:

```
$ git checkout main
```

```
$ git merge feature-1 --ff
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git checkout main
Switched to branch 'main'
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git merge feature-1 --ff
Updating beaa8f1..61a7fb0
Fast-forward
  version.txt | 1 +
   1 file changed, 1 insertion(+)
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ 
```

3.

Fusiona feature-2 usando non-fast-forward:

```
$ git merge feature-2 --no-ff
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git merge feature-2 --no-ff
Auto-merging version.txt
CONFLICT (content): Merge conflict in version.txt
Automatic merge failed; fix conflicts and then commit the result.
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ code .
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git add .
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git commit -m "resolver conflicts and merge with feature-2"
[main b987eb8] resolver conflicts and merge with feature-2
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git log --oneline
b987eb8 (HEAD -> main) resolver conflicts and merge with feature-2
db8c928 (feature-2) Se agrega caracteristica 2
61a7fb0 (feature-1) Agregar caracteristica 1
beaa8f1 ....
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git log --graph --oneline
*   b987eb8 (HEAD -> main) resolver conflicts and merge with feature-2
|\ 
| * db8c928 (feature-2) Se agrega caracteristica 2
* | 61a7fb0 (feature-1) Agregar caracteristica 1
|/
* beaa8f1 ....
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ 
```

# 4.

Realiza una nueva rama feature-3 con múltiples commits y fúnsionala con squash:

```
$ git checkout -b feature-3  
$ echo "Caracteristica 3 paso 1" >> version.txt  
$ git add version.txt  
$ git commit -m "Caracteristica 3 paso 1"  
$ echo "Caracteristica 3 paso 2" >> version.txt  
$ git add version.txt  
$ git commit -m "Caracteristica 3 paso 2"  
$ git checkout main  
$ git merge --squash feature-3  
$ git commit -m "Aregar caracteristica 3 en un commit"
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git checkout -b feature-3  
Switched to a new branch 'feature-3'  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ echo "Caracteristica 3 paso 1" >> version.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git add version.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git commit -m "Caracteristica 3 paso 1"  
[feature-3 2322a40] Caracteristica 3 paso 1  
 1 file changed, 1 insertion(+)  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ echo "Caracteristica 3 paso 2" >> version.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git add version.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git commit -m "Caracteristica 3 paso 2"  
[feature-3 6a7074f] Caracteristica 3 paso 2  
 1 file changed, 1 insertion(+)  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git checkout main  
Switched to branch 'main'  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git merge --squash feature-3  
Updating b987eb8..6a7074f  
Fast-forward  
Squash commit -- not updating HEAD  
 version.txt | 2 ++  
 1 file changed, 2 insertions(+)  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git commit -m "Aregar caracteristica 3 en un commit"  
[main 8675181] Agregar caracteristica 3 en un commit  
 1 file changed, 2 insertions(+)
```

## 5. Compara el historial de Git:

Historial Fast-forward:

```
$ git log --graph --oneline --first-parent
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git log --graph --oneline --first-parent
* 8675181 (HEAD -> main) Agregar caracteristica 3 en un commit
* b987eb8 resolver conflicts and merge with feature-2
* 61a7fb0 (feature-1) Agregar caracteristica 1
* beaa8f1 ....
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$
```

Historial Non-fast-forward:

```
$ git log --graph --oneline -merges
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git log --graph --oneline -merges
fatal: ambiguous argument '-merges': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$
```

Historial con Squash:

```
$ git log --graph --oneline --decorate --all
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$ git log --graph --oneline --decorate --all
* 8675181 (HEAD -> main) Agregar caracteristica 3 en un commit
| * 6a7074f (feature-3) Caracteristica 3 paso 2
| * 2322a40 Caracteristica 3 paso 1
|/
* b987eb8 resolver conflicts and merge with feature-2
|\ 
| * db8c928 (feature-2) Se agrega caracteristica 2
* | 61a7fb0 (feature-1) Agregar caracteristica 1
|/
* beaa8f1 ....
diegodev@HPavilion:~/Desktop/dev-practice/prueba-compare-merge$
```

# PREGUNTAS

¿Cómo se ve el historial en cada tipo de fusión?

En el caso de Non-fast-forward y Squash se ve un commit extra de fusion, que indica el punto de integracion a la rama main, pero en el caso de Fast Forward no tenemos ese commit extra.

¿Qué método prefieres en diferentes escenarios y por qué?

Fast-forward para proyectos pequeños, donde sepamos de antemano que no tendremos conflictos para mantener un historial limpio. Non-fast-forward para trabajos colaborativos donde se muestre la trazabilidad del proyecto, mostrando el punto de integracion de la fusion de las ramas. Y Squash para juntar varios commits pequeños en uno solo que represente de manera clara en el main.

# EJERCICIO: USANDO FUSIONES AUTOMÁTICAS Y REVERTIR FUSIONES

En este ejercicio, aprenderás cómo Git puede fusionar automáticamente cambios cuando no hay conflictos y cómo revertir una fusión si cometes un error.

# 1.

Inicializa un nuevo repositorio y realiza dos commits en main:

```
$ mkdir prueba-auto-merge  
$ cd prueba-auto-merge  
$ git init  
$ echo "Linea 1" > file.txt  
$ git add file.txt  
$ git commit -m "Agrega linea 1"  
$ echo "Linea 2" >> file.txt  
$ git add file.txt  
$ git commit -m "...linea 2"
```

```
diegodev@HPavilion:~/Desktop/dev-practice$ mkdir prueba-auto-merge  
diegodev@HPavilion:~/Desktop/dev-practice$ cd prueba-auto-merge  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git init  
Initialized empty Git repository in /home/diegodev/Desktop/dev-practice/prueba-auto-merge/.git/  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ echo "Linea 1" > file.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git add file.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git commit -m "Agrega linea 1"  
[main (root-commit) 5cc7eb1] Agrega linea 1  
 1 file changed, 1 insertion(+)  
  create mode 100644 file.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ echo "Linea 2" >> file.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git add file.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git commit -m "agrega linea 2"  
[main 60107e7] agrega linea 2  
 1 file changed, 1 insertion(+)  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$
```

## 2.

Crea una nueva rama auto-merge y realiza otro commit en file.txt:

```
git checkout -b auto-merge  
echo "Linea 3" >> file.txt  
git add file.txt  
git commit -m "... linea 3"
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git checkout -b auto-merge  
Switched to a new branch 'auto-merge'  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ echo "Linea 3" >> file.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git add file.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git commit -m "agrega linea 3"  
[auto-merge bc67e00] agrega linea 3  
 1 file changed, 1 insertion(+)
```

## 3.

Vuelve a main y realiza cambios no conflictivos en otra parte del archivo:

```
git checkout main  
echo "Footer: Fin del archivo" >> file.txt  
git add file.txt  
git commit -m "Add footer al archivo file.txt"
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git checkout main  
Switched to branch 'main'  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ nano file  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ nano file.txt  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ cat file.txt  
Linea 1 cabecera  
Linea 2  
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$
```

## 4.

Fusiona la rama auto-merge con main:

```
git merge auto-merge
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git merge auto-merge
Auto-merging file.txt
Merge made by the 'ort' strategy.
  file.txt | 1 +
  1 file changed, 1 insertion(+)
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ □
```

## 5.

Git debería fusionar los cambios automáticamente sin conflictos.

```
GNU nano 6.2          /home/diegodev/Desktop/dev-practice/prueba-auto-merge/.git/MERGE_MSG
Merge branch 'auto-merge'
# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git log --graph --oneline
*   868d796 (HEAD -> main) Merge branch 'auto-merge'
|\ 
| * bc67e00 (auto-merge) agrega linea 3
* | a30ec83 agrega cabecera
|/
* 60107e7 agrega linea 2
* 5cc7eb1 Agrega linea 1
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ □
```

## 6.

Revertir la fusión: Si decides que la fusión fue un error, puedes revertirla:

```
git revert -m 1 HEAD
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git revert -m 1 HEAD
[main b6fcada] Revert "Merge branch 'auto-merge'"
 1 file changed, 1 deletion(-)
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ □
```

## 7.

Verifica el historial:

```
git log --graph --oneline
```

```
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ git log --graph --oneline
* b6fcada (HEAD -> main) Revert "Merge branch 'auto-merge'"
*   868d796 Merge branch 'auto-merge'
|\ 
| * bc67e00 (auto-merge) agrega linea 3
* | a30ec83 agrega cabecera
|/
* 60107e7 agrega linea 2
* 5cc7eb1 Agrega linea 1
diegodev@HPavilion:~/Desktop/dev-practice/prueba-auto-merge$ □
```

# PREGUNTAS

- ¿Cuándo usarías un comando como git revert para deshacer una fusión?

Cuando realice una fusión que no es correcta, es decir al combinar las ramas, cometí errores entonces puedo revertir los cambios de la fusión sin alterar el historial, ya que esto sería grave en trabajos colaborativos.

- ¿Qué tan útil es la función de fusión automática en Git?

Es muy útil cuando fusionamos ramas que no interfieren entre sí, entonces sabemos de antemano que esto no generará conflictos entonces dejamos que se dé la fusión automática de git para agilizar el proceso.

# EJERCICIO: FUSIÓN REMOTA EN UN REPOSITORIO COLABORATIVO

Este ejercicio te permitirá practicar la fusión de ramas en un entorno remoto colaborativo, simulando un flujo de trabajo de equipo.

1. Clona un repositorio remoto desde GitHub o crea uno nuevo:

```
$ git clone https://github.com/tu-usuario/nombre-del-repositorio.git  
$ cd nombre-del-repositorio
```

```
diegodev@HPavilion:~/Desktop/dev-practice$ mkdir FusionRemota  
diegodev@HPavilion:~/Desktop/dev-practice$ cd FusionRemota/  
diegodev@HPavilion:~/Desktop/dev-practice/FusionRemota$ echo "# Fusion-Remota" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin git@github.com:0x-Chema-x0/Fusion-Remota.git  
git push -u origin main  
Initialized empty Git repository in /home/diegodev/Desktop/dev-practice/FusionRemota/.git/  
[main (root-commit) a6b8169] first commit  
 1 file changed, 1 insertion(+)  
 create mode 100644 README.md  
Enumerating objects: 3, done.  
Counting objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 227 bytes | 227.00 KiB/s, done.  
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
To github.com:0x-Chema-x0/Fusion-Remota.git  
 * [new branch]      main -> main  
Branch 'main' set up to track remote branch 'main' from 'origin'.  
diegodev@HPavilion:~/Desktop/dev-practice/FusionRemota$ █
```

2. Crea una nueva rama colaboracion y haz algunos cambios:

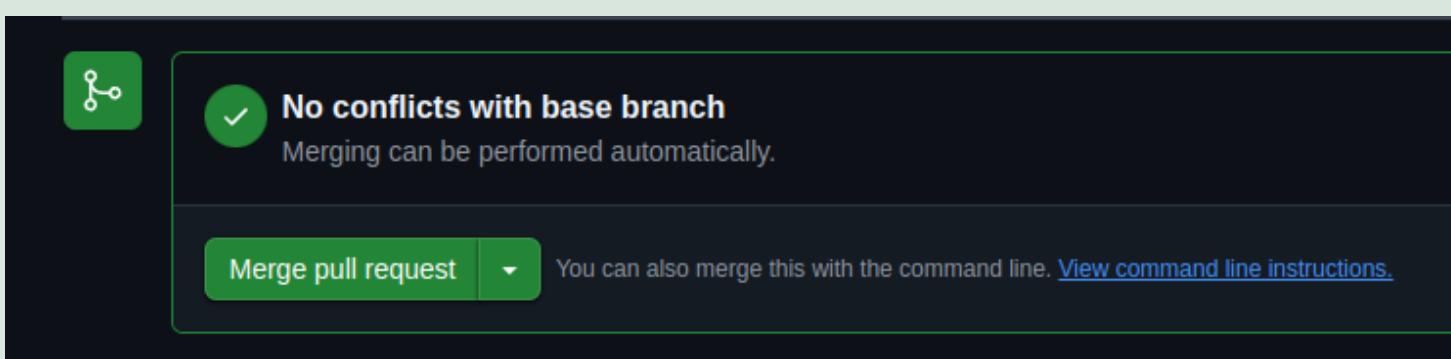
```
$ git checkout -b colaboracion  
$ echo "Colaboración remota" > colaboracion.txt  
$ git add colaboracion.txt  
$ git commit -m "add colaboracion.txt"
```

```
diegodev@HPavilion:~/Desktop/dev-practice/FusionRemota$ git checkout -b colaboracion  
Switched to a new branch 'colaboracion'  
diegodev@HPavilion:~/Desktop/dev-practice/FusionRemota$ echo "Colaboración remota" > colaboracion.txt  
diegodev@HPavilion:~/Desktop/dev-practice/FusionRemota$ git add colaboracion.txt  
diegodev@HPavilion:~/Desktop/dev-practice/FusionRemota$ git commit -m "add colaboracion file"  
[colaboracion c65e72f] add colaboracion file  
 1 file changed, 1 insertion(+)  
 create mode 100644 colaboracion.txt  
diegodev@HPavilion:~/Desktop/dev-practice/FusionRemota$ █
```

- 3.** Empuja los cambios a la rama remota:  
\$ git push origin colaboracion

```
diegodev@HPavilion:~/Desktop/dev-practice/FusionRemota$ git push origin colaboracion
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), 308 bytes | 308.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'colaboracion' on GitHub by visiting:
remote:     https://github.com/Ox-Chema-xO/Fusion-Remota/pull/new/colaboracion
remote:
To github.com:Ox-Chema-xO/Fusion-Remota.git
 * [new branch]      colaboracion -> colaboracion
diegodev@HPavilion:~/Desktop/dev-practice/FusionRemota$
```

- 4.** Simula una fusión desde la rama colaboracion en la rama main de otro colaborador. (Puedes usar la interfaz de GitHub para crear un Pull Request y realizar la fusión).



The image contains three screenshots related to GitHub pull requests:

- Top Screenshot:** A "Create a pull request" dialog box. It shows the "base: main" and "compare: colaboracion" dropdowns, both set to their respective branches. A green checkmark indicates "Able to merge. These branches can be automatically merged." Fields for "Add a title" (containing "add colaboracion file") and "Add a description" (with a rich text editor) are present. On the right, there are sections for "Reviewers" (no reviews), "Assignees" (no assignees), "Labels" (none yet), "Projects" (none yet), "Milestone" (no milestone), "Development" (use closing keywords), and "Helpful resources" (GitHub Community Guidelines). A "Create pull request" button is at the bottom.
- Middle Screenshot:** A "Merge pull request #1 from Ox-Chema-xO/colaboracion" screen. It shows the commit history: "b5e1e41 · now · 3 Commits". The commits are: "README.md" (first commit, 10 minutes ago) and "colaboracion.txt" (add colaboracion file, 6 minutes ago). Below the commits, there is a "README" section.
- Bottom Screenshot:** A "Fusion-Remota" repository page. It shows the repository name and a brief description.

# PREGUNTAS

¿Cómo cambia la estrategia de fusión cuando colaboras con otras personas en un repositorio remoto?

La estrategia cambia porque antes de fusionar debe ser consultado con el equipo, seguir lo establecido o acordado, de acuerdo a ello se acepta la fusión.

¿Qué problemas comunes pueden surgir al integrar ramas remotas?

Pueden surgir conflictos de fusión debido a varias modificaciones en el mismo archivo, se puede llegar a tener un exceso de commits perjudicando la trazabilidad y claridad del historial de commits.

# EJERCICIO FINAL: SIMULACION DE FLUJO DE TRABAJO COMPLETO

# CONFIGURA UN PROYECTO SIMULADO:

1. Crea un proyecto con tres ramas: main, feature1, y feature2.

```
diegodev@HPavilion:~/Desktop/dev-practice$ mkdir simulacion-proyecto
diegodev@HPavilion:~/Desktop/dev-practice$ cd simulacion-proyecto/
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git init
Initialized empty Git repository in /home/diegodev/Desktop/dev-practice/simulacion-proyecto/.git/
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ echo "# Simulacion de un proyecto colaborativo" > README.md
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git add README.md
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git commit -m "add readme file"
[main (root-commit) 8c87e60] add readme file
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git branch feature1
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git branch feature2
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git log --oneline
8c87e60 (HEAD -> main, feature2, feature1) add readme file
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ █
```

# CONFIGURA UN PROYECTO SIMULADO:

2. Realiza varios cambios en feature1 y feature2 y simula colaboraciones paralelas.

## feature1:

```
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ git checkout feature1
Switched to branch 'feature1'
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ echo "Funcionalidad 1 - paso 1" > f1.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ git add f1.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ git commit -m "add feature1 with primer paso"
[feature1 7d44807] add feature1 with primer paso
 1 file changed, 1 insertion(+)
  create mode 100644 f1.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ echo "Funcionalidad 1 - paso 2" >> f1.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ git add .
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ git commit -m "add feature1 with segundo paso"
[feature1 77c1a61] add feature1 with segundo paso
 1 file changed, 1 insertion(+)
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$
```

## feature2:

```
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ git checkout feature2
Switched to branch 'feature2'
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ echo "Funcionalidad 2 - parte A" > f2.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ git add f2.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ git commit -m "add feature2 with part A"
[feature2 d97a8c2] add feature2 with part A
 1 file changed, 1 insertion(+)
  create mode 100644 f2.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ echo "Funcionalidad 2 - parte B" >> f2.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ git add f2.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$ git commit -m "add feature2 with part B"
[feature2 47fcfba] add feature2 with part B
 1 file changed, 1 insertion(+)
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-projecto$
```

# CONFIGURA UN PROYECTO SIMULADO:

## 3. Realiza fusiones utilizando diferentes métodos:

- Fusiona feature1 con main utilizando git merge --ff.

```
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git checkout main
Switched to branch 'main'
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git merge --ff feature1
Updating 8c87e60..77c1a61
Fast-forward
 f1.txt | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 f1.txt
```

- Fusiona feature2 con main utilizando git merge --no-ff.

```
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git merge --no-ff feature2 -m "merge with feature2 usando --no-ff"
"
Merge made by the 'ort' strategy.
 f2.txt | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 f2.txt
```

- Haz una rama adicional llamada feature3 y aplasta sus commits utilizando git merge --squash.

```
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git checkout -b feature3
Switched to a new branch 'feature3'
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ echo "Funcionalidad 3 - paso 1" > f3.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git add f3.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git commit -m "add feature3 with primer paso"
[feature3 0fec6eb] add feature3 with primer paso
 1 file changed, 1 insertion(+)
 create mode 100644 f3.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ echo "Funcionalidad 3 - paso 2" >> f3.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git add f3.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git commit -m "add feature3 with segundo paso"
[feature3 7184ddd] add feature3 with segundo paso
 1 file changed, 1 insertion(+)
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git checkout main
Switched to branch 'main'
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git merge --squash feature3
Updating 42269e9..7184ddd
Fast-forward
Squash commit -- not updating HEAD
 f3.txt | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 f3.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git commit -m "Feature3 fusionada con squash"
[main a717019] Feature3 fusionada con squash
 1 file changed, 2 insertions(+)
 create mode 100644 f3.txt
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$
```

# CONFIGURA UN PROYECTO SIMULADO:

## 4. Analiza el historial de commits:

- Revisa el historial para entender cómo los diferentes métodos de fusión afectan el árbol de commits.
- Compara los resultados y discute con tus compañeros de equipo cuál sería la mejor estrategia de fusión para proyectos más grandes.

```
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$ git log --oneline --graph --all
* a717019 (HEAD -> main) Feature3 fusionada con squash
| * 7184ddd (feature3) add feature3 with segundo paso
| * 0fec6eb add feature3 with primer paso
|/
* 42269e9 merge with feature2 usando --no-ff
|\ 
| * 47fcfba (feature2) add feature2 with part B
| * d97a8c2 add feature2 with part A
* | 77c1a61 (feature1) add feature1 with segundo paso
* | 7d44807 add feature1 with primer paso
|/
* 8c87e60 add readme file
diegodev@HPavilion:~/Desktop/dev-practice/simulacion-proyecto$
```

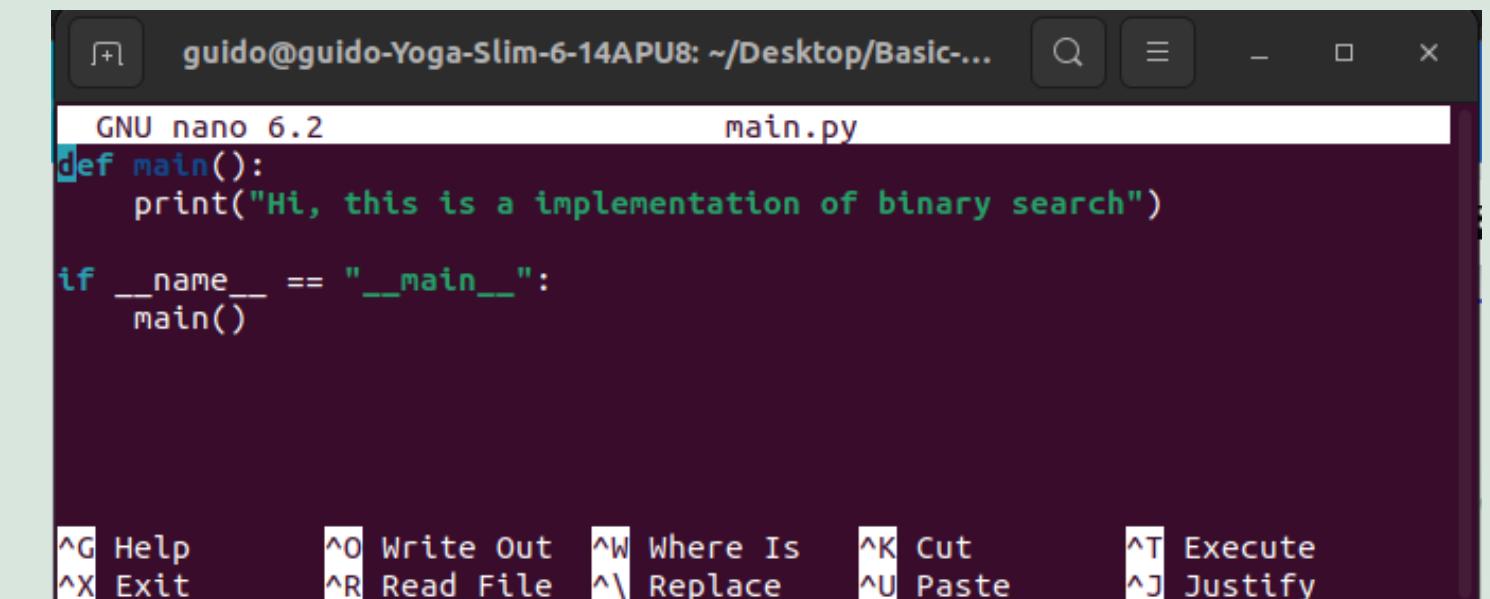
La mejor estrategia de fusion para proyectos más grandes seria no-fast-forward debido a que nos permite tener ese commit de fusion, al fusionar ramas, mostrando perfectamente el punto de integracion, sabiendo que ramas se fusionaron , cuando y porque todo, ademas no debemos hacer un uso excesivo de commits de fusion ya que satura el historial impidiendo su comprendimiento.

# EJERCICIO FINAL: FLUJO DE TRABAJO COMPLETO EN EQUIPO

# CONFIGURA UN PROYECTO SIMULADO:

1. Crea un proyecto con tres ramas: main, feature1, y feature2.

```
diegodev@HPavilion:~/Desktop/dev-practice$ cd BinarySearch/
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git init
Initialized empty Git repository in /home/diegodev/Desktop/dev-practice/BinarySearch/.git/
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ echo "# Project about binary search" > README.md
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ touch main.py
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ code .
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git status
On branch main
No commits yet
Untracked files:
(use "git add <file>..." to include in what will be committed)
 README.md
 main.py
nothing added to commit but untracked files present (use "git add" to track)
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git add .
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git commit -m "Base structure"
[main (root-commit) 149d484] Base structure
 2 files changed, 6 insertions(+)
  create mode 100644 README.md
  create mode 100644 main.py
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git branch feature1
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git branch feature2
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git branch
  feature1
  feature2
* main
```



```
GNU nano 6.2          main.py
def main():
    print("Hi, this is a implementation of binary search")

if __name__ == "__main__":
    main()

^G Help      ^O Write Out  ^W Where Is  ^K Cut
^X Exit      ^R Read File   ^V Replace  ^U Paste
^T Execute    ^J Justify
```

```
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git remote add origin git@github.com:0x-Chema-x0/Basic-BinarySearch.git
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 373 bytes | 373.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:0x-Chema-x0/Basic-BinarySearch.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git push --all origin
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:0x-Chema-x0/Basic-BinarySearch.git
 * [new branch]      feature1 -> feature1
 * [new branch]      feature2 -> feature2
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git push --tags origin
Everything up-to-date
```

# CONFIGURA UN PROYECTO SIMULADO:

2. Realiza varios cambios en feature1 y feature2 y simula colaboraciones paralelas.

## feature1:

```
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git checkout feature1
Switched to branch 'feature1'
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ code .
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git status
On branch feature1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   main.py

no changes added to commit (use "git add" and/or "git commit -a")
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git commit -m "add bubble-sort for sorted the array"
[feature1 a4690e0] add bubble-sort for sorted the array
 1 file changed, 14 insertions(+), 2 deletions(-)
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ 
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ 
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git status
On branch feature1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   main.py

no changes added to commit (use "git add" and/or "git commit -a")
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git add main.py
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git commit -m "bubble-sort optimizado para ordenar el arreglo"
[feature1 3f057f3] bubble-sort optimizado para ordenar el arreglo
 1 file changed, 5 insertions(+), 1 deletion(-)
```

```
1 def bubble_sort(arr):
2     n = len(arr)
3     for i in range(n):
4         swap = False
5         for j in range(n-i-1):
6             if arr[j] > arr[j+1]:
7                 arr[j], arr[j+1] = arr[j+1], arr[j]
8                 swap = True
9             if not swap:
10                 break
11     return arr
```

## feature2:

```
guido@guido-Yoga-Slim-6-14APU8: ~/Desktop/Basic-BinarySearch
GNU nano 6.2
main.py
def binary_search(arr, target):
    """
    Implementación de búsqueda binaria.
    Retorna el índice del elemento si se encuentra, -1 si no.
    """
    left = 0
    right = len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        # Si el elemento es el del medio, lo encontramos
        if arr[mid] == target:
            return mid
        # Si el elemento es mayor, ignoramos la mitad izquierda
        elif arr[mid] < target:
            left = mid + 1
        # Si el elemento es menor, ignoramos la mitad derecha
        else:
            right = mid - 1
        # El elemento no está presente
        return -1
def main():
    print("Hi, this is a implementation of binary search")
    sorted_list = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
    print(f"Lista ordenada: {sorted_list}")
    # Buscar varios elementos
    test_cases = [16, 72, 42]
    for target in test_cases:
        result = binary_search(sorted_list, target)
        if result != -1:
            print(f"El elemento {target} está en el indice {result}")
        else:
            print(f"El elemento {target} no está en la lista")
if __name__ == "__main__":
    main()
^G Help      ^O Write Out   ^W Where Is   ^K Cut       ^T Execute   ^C Location   M-U Undo
^X Exit      ^R Read File   ^M Replace   ^U Paste    ^J Justify   ^I Go To Line  M-E Redo
^D Open to up to see more or right/reload :
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git add main.py
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git commit -m "add binary_search in main.py"
[feature2 ad39791] add binary_search in main.py
 1 file changed, 37 insertions(+), 1 deletion(-)
autodocido visto clínica 14apu8:~/Desktop/Basic-BinarySearch$ git status
```

# CONFIGURA UN PROYECTO SIMULADO:

## 3. Realiza fusiones utilizando diferentes métodos:

- Fusiona feature1 con main utilizando git merge --ff.

```
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git merge feature1
Updating 149d484..3f057f3
Fast-forward
 main.py | 20 ++++++-----+
 1 file changed, 18 insertions(+), 2 deletions(-)
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 855 bytes | 855.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To github.com:0x-Chema-x0/Basic-BinarySearch.git
 149d484..3f057f3  main -> main
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$
```

- Haz una rama adicional llamada feature3 y aplasta sus commits utilizando git merge --squash.

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git checkout -b feature3
Switched to a new branch 'feature3'
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ code .
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git add main.py
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git commit -m "add execution time"
[feature3 f7216b5] add execution time
 1 file changed, 6 insertions(+)
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git merge --squash
Already up to date. (nothing to squash)
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git merge --squash feature3
Updating e40b1b5..f7216b5
Fast-forward
Squash commit -- not updating HEAD
 main.py | 6 ++++++
 1 file changed, 6 insertions(+)
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git commit -m "Añadi ejecucion de tiempo a main.py"
[main e0553fa] Añadi ejecucion de tiempo a main.py
 1 file changed, 6 insertions(+)
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git push
Username for 'https://github.com': GuidoCh23
Password for 'https://GuidoCh23@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.42 KiB | 1.42 MiB/s, done.
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/0x-Chema-x0/Basic-BinarySearch.git
 3f057f3..e40b1b5  main -> main
```

- Fusiona feature2 con main utilizando git merge --no-ff.

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git merge --no-ff feature2
Merge made by the 'ort' strategy.
 main.py | 38 ++++++-----+
 1 file changed, 37 insertions(+), 1 deletion(-)
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ fatal: Need to specify how to reconcile divergent branches.
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git config pull.rebase false
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git pull
Auto-merging main.py
CONFLICT (content): Merge conflict in main.py
Automatic merge failed; fix conflicts and then commit the result.
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git branch
 * feature2
* main
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ nano main.py
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git status
On branch main
Your branch and 'origin/main' have diverged,
and have 2 and 2 different commits each, respectively.
 (use "git pull" to merge the remote branch into yours)

You have unmerged paths.
 (fix conflicts and run "git commit")
 (use "git merge --abort" to abort the merge)

Unmerged paths:
 (use "git add <file>..." to mark resolution)
 both modified: main.py

no changes added to commit (use "git add" and/or "git commit -a")
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git add main.py
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git commit -m "Conflict resuelto en main.py"
[main e40b1b5] Conflicto resuelto en main.py
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/Basic-BinarySearch$ git push
Username for 'https://github.com': GuidoCh23
Password for 'https://GuidoCh23@github.com':
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.42 KiB | 1.42 MiB/s, done.
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/0x-Chema-x0/Basic-BinarySearch.git
 3f057f3..e40b1b5  main -> main
```

# CONFIGURA UN PROYECTO SIMULADO:

4.

Analiza el historial de commits:

- Revisa el historial para entender cómo los diferentes métodos de fusión afectan el árbol de commits.

- Compara los resultados y discute con tus compañeros de equipo cuál sería la mejor estrategia de fusión para proyectos más grandes.

```
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git pull origin main
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 10 (delta 3), reused 10 (delta 3), pack-reused 0 (from 0)
Unpacking objects: 100% (10/10), 1.81 KiB | 370.00 KiB/s, done.
From github.com:Ox-Chema-x0/Basic-BinarySearch
 * branch           main      -> FETCH_HEAD
   3f057f3..e0553fa main      -> origin/main
Updating 3f057f3..e0553fa
Fast-forward
 main.py | 43 ++++++-----+
 1 file changed, 38 insertions(+), 5 deletions(-)
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$ git log --oneline --graph --all
* e0553fa (HEAD -> main, origin/main) Añadi ejecucion de tiempo a main.py
* e40b1b5 Conflicto resuelto en main.py
|\ 
| * 3f057f3 (origin/feature1, feature1) bubble-sort optimizado para ordenar el arreglo
| * a4690e0 add bubble-sort for sorted the array
* | 75ecc7e Merge branch 'feature2'
|/ 
|/ 
| * ad39791 add binary_search in main.py
|
* 149d484 (origin/feature2, feature2) Base structure
diegodev@HPavilion:~/Desktop/dev-practice/BinarySearch$
```

La mejor estrategia de fusion para proyectos más grandes seria no-fast-forward debido a que nos permite tener ese commit de fusion, al fusionar ramas, mostrando perfectamente el punto de integracion, sabiendo que ramas se fusionaron , cuando y porque todo, ademas no debemos hacer un uso excesivo de commits de fusion ya que satura el historial impidiendo su comprendimiento.

MUCHAS  
GRACIAS