



DESARROLLO DE SOFTWARE

ACTIVIDAD 4

Aarón Flores Alberca

Guido Chipana Calderon

Diego Delgado Velarde



Facultad de ciencias – Universidad Nacional de Ingeniería

EJERCICIO 1



OBJETIVO

Practicar la creacion, fusion y eliminacion de ramas, así como la resolucion de conflictos que puedan surgir durante la fusion.

Crear una nueva rama para una característica:

- Crea una nueva rama llamada feature/advanced-feature desde la rama main:

```
$ git branch feature/advanced-feature
```

```
$ git checkout feature/advanced-feature
```

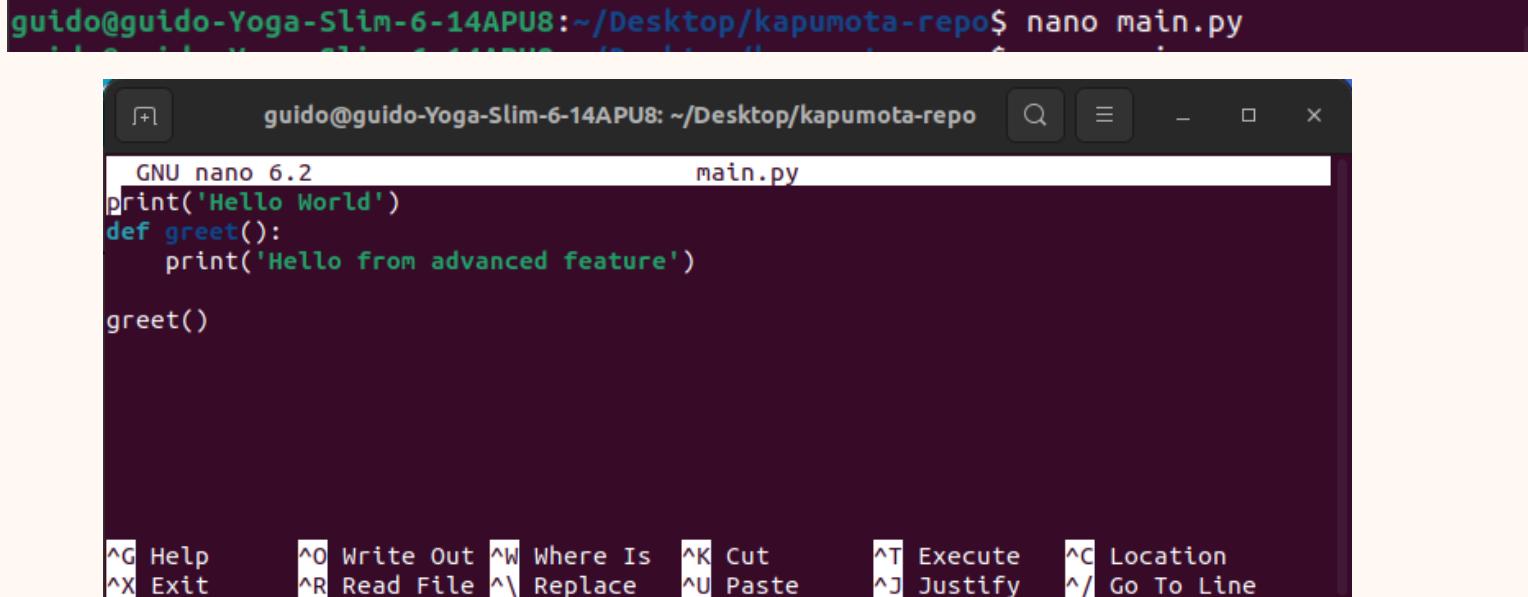
```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git branch feature/advanced-feature
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git checkout feature/advanced-feature
Switched to branch 'feature/advanced-feature'
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git branch
  develop
* feature/advanced-feature
  feature/another-new-feature
  feature/login
  hotfix/bugfix
  master
```

Modificar archivos en la nueva rama:

- Edita el archivo main.py para incluir una función adicional:

```
def greet():
    print('Hello from advanced feature')

greet()
```



```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ nano main.py
```

```
GNU nano 6.2
main.py
print('Hello World')
def greet():
    print('Hello from advanced feature')

greet()

^G Help      ^O Write Out  ^W Where Is  ^K Cut      ^T Execute  ^C Location
^X Exit      ^R Read File  ^L Replace   ^U Paste    ^J Justify  ^/ Go To Line
```

- Añade y confirma estos cambios en la rama feature/advanced-feature:

```
$ git add main.py
```

```
$ git commit -m "Add greet function in advanced feature"
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git status
On branch feature/advanced-feature
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

guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git commit -m "Add greet function in advanced feature"
[feature/advanced-feature 4f4a679] Add greet function in advanced feature
 1 file changed, 4 insertions(+)
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git status
On branch feature/advanced-feature
nothing to commit, working tree clean
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git log --oneline
4f4a679 (HEAD -> feature/advanced-feature) Add greet function in advanced feature
19474a5 (master, feature/another-new-feature, develop) add main.py
7c10371 Set up the repository base documentation
cf05b8d Initial commit with README.md
```

Simular un desarrollo paralelo en la rama master:

- Cambia de nuevo a la rama master:

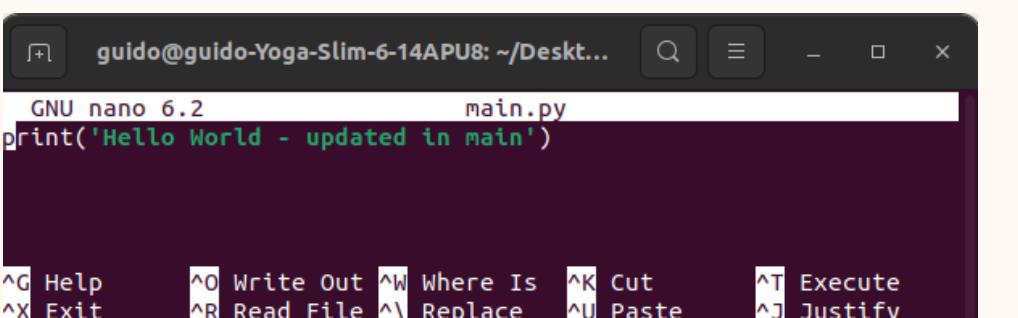
```
$ git checkout master
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git checkout master
Switched to branch 'master'
```

- Edita el archivo main.py de forma diferente (por ejemplo, cambia el mensaje del print original):

```
print('Hello World - updated in main')
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ nano main.py
```



- Añade y confirma estos cambios en la rama master:

```
$ git add main.py
```

```
$ git commit -m "Update main.py message in master branch"
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git add main.py
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git commit -m "Update main.py message in master branch"
[master 01d7d20] Update main.py message in master branch
 1 file changed, 1 insertion(+), 5 deletions(-)
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git status
On branch master
nothing to commit, working tree clean
```

Intentar fusionar la rama feature/advanced-feature en main:

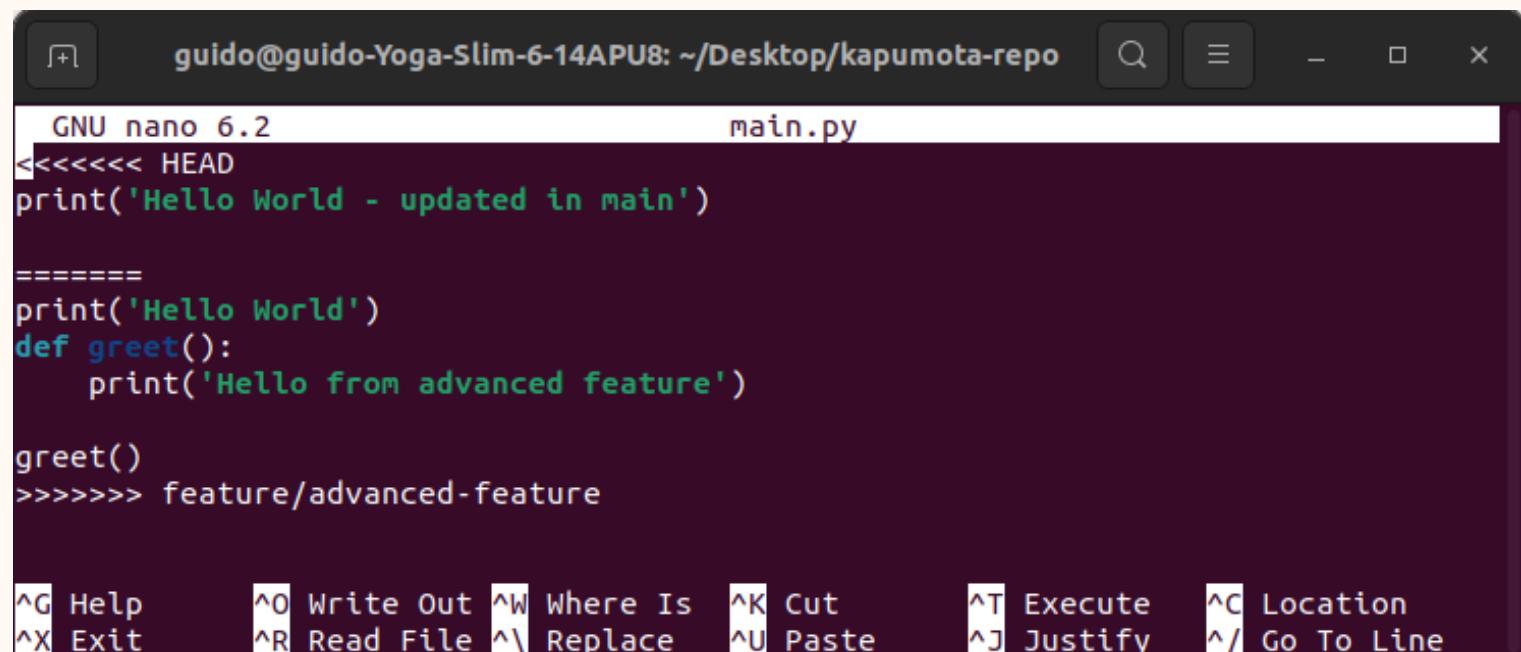
- Fusiona la rama feature/advanced-feature en master:

```
$ git merge feature/advanced-feature
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git merge feature/advanced-feature
Auto-merging main.py
CONFLICT (content): Merge conflict in main.py
Automatic merge failed; fix conflicts and then commit the result.
```

Resolver el conflicto de fusión:

- Git generará un conflicto en main.py. Abre el archivo y resuelve el conflicto manualmente, eligiendo cómo combinar las dos versiones.

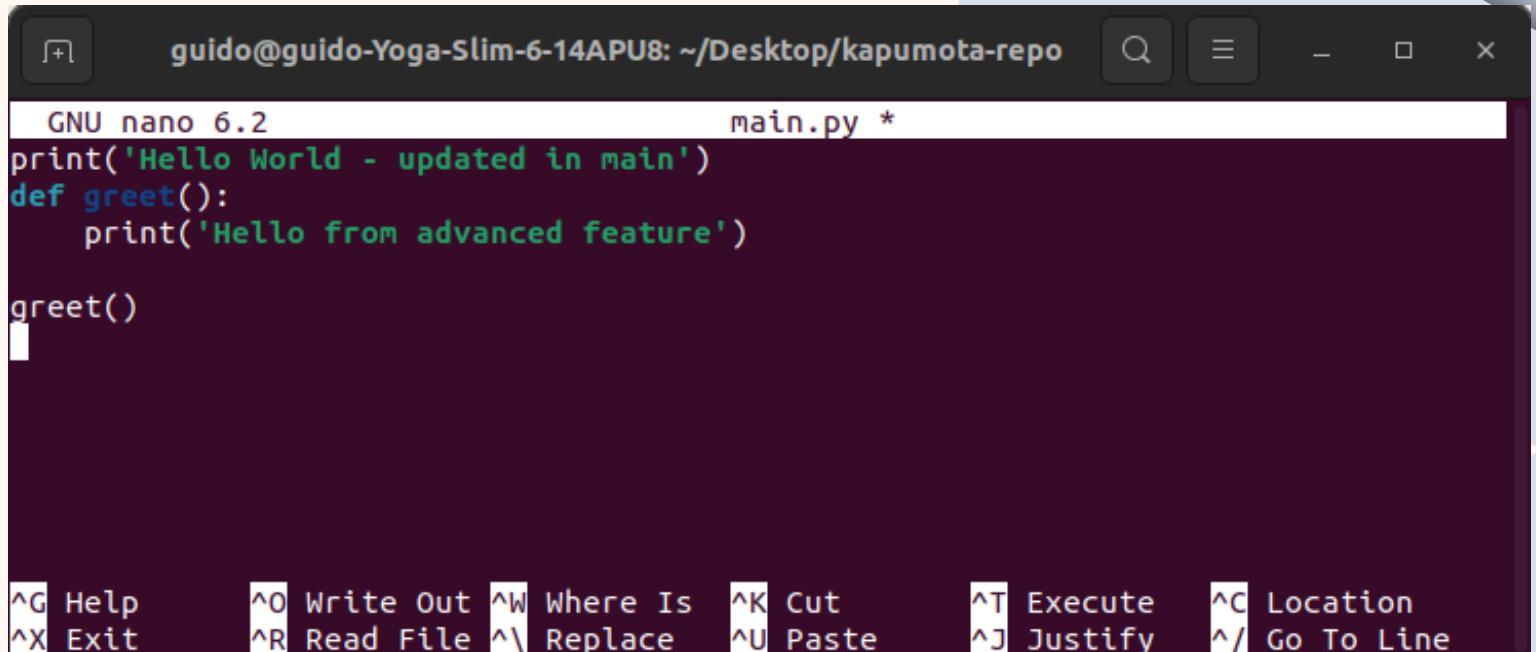


```
guido@guido-Yoga-Slim-6-14APU8: ~/Desktop/kapumota-repo
GNU nano 6.2
main.py
<<<< HEAD
print('Hello World - updated in main')

=====
print('Hello World')
def greet():
    print('Hello from advanced feature')

greet()
>>>> feature/advanced-feature

^G Help      ^O Write Out  ^W Where Is  ^K Cut      ^T Execute  ^C Location
^X Exit      ^R Read File  ^\ Replace   ^U Paste    ^J Justify  ^/ Go To Line
```



```
guido@guido-Yoga-Slim-6-14APU8: ~/Desktop/kapumota-repo
main.py *
GNU nano 6.2
print('Hello World - updated in main')
def greet():
    print('Hello from advanced feature')

greet()
```

- Despues de resolver el conflicto, añade el archivo resuelto y completa la fusión:

```
$ git add main.py
```

```
$ git commit -m "Resolve merge conflict between main and feature/advanced-feature"
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git add main.py
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git commit -m "Resolve merge conflict between master and feature/advanced-feature"
[master fe34bde] Resolve merge conflict between master and feature/advanced-feature
```

Eliminar la rama fusionada:

- Una vez que hayas fusionado con éxito y resuelto los conflictos, elimina la rama feature/advanced-feature:

```
$ git branch -d feature/advanced-feature
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git branch -d feature/advanced-feature
Deleted branch feature/advanced-feature (was 4f4a679).
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git branch
  develop
  feature/another-new-feature
  feature/login
  hotfix/bugfix
* master
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$
```

EJERCICIO 2



OBJETIVO

Aprender a navegar y manipular el historial de commits usando comandos avanzados de Git

Se tiene un repositorio de git en el que se almacenan archivos Markdown para Obsidian

The screenshot shows the Obsidian application interface. On the left, there is a file tree with the following structure:

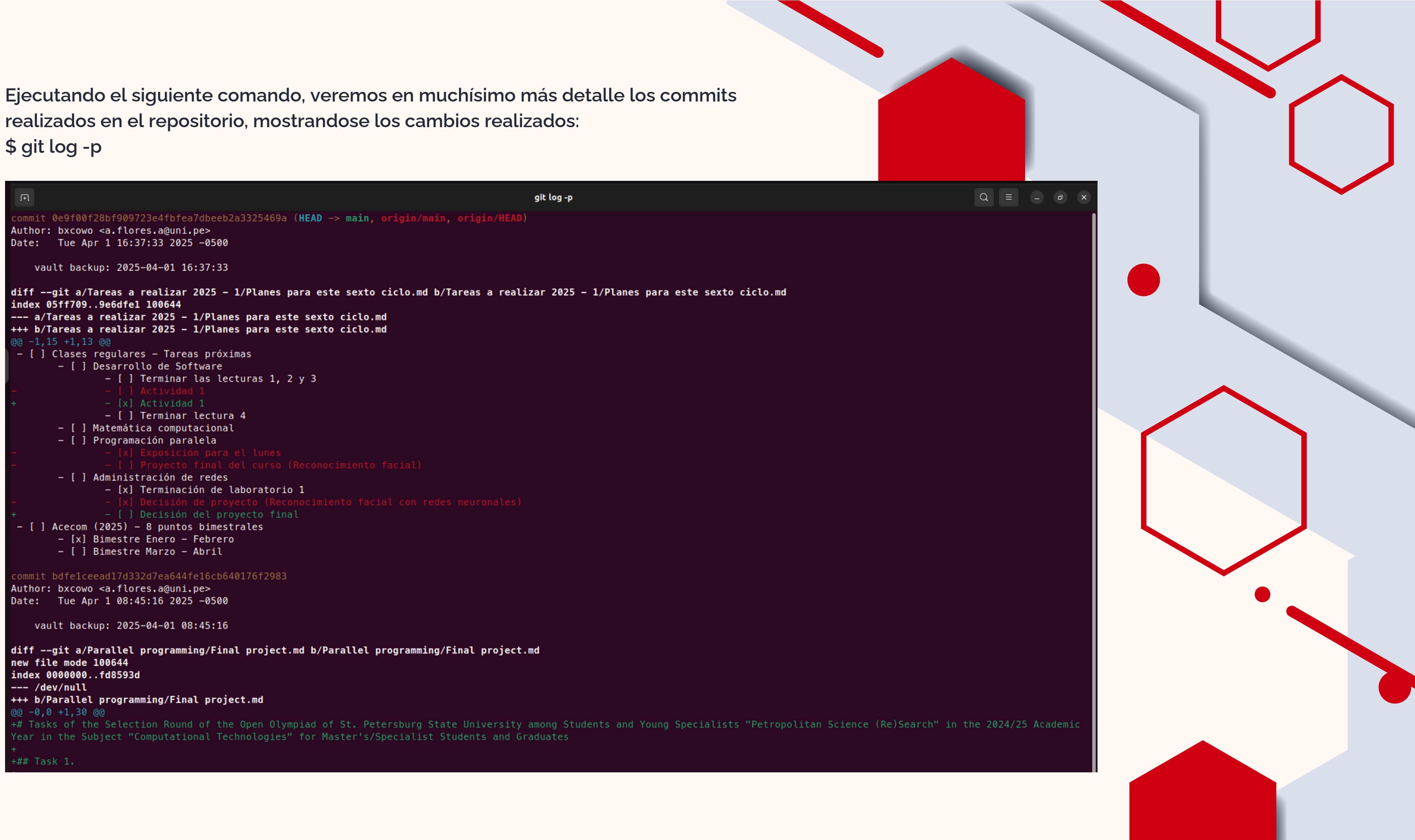
- > Computational Mathematics
- > Google Cloud Computing Foundations Acad...
- > knowledge
- Parallel programming
 - Final project
- Software Development
 - Actividad 1
 - Actividad 2
 - Lectura1
 - Lectura2
 - Lectura3
 - Lectura4
 - Lectura5
 - Lectura6
 - Lectura7
 - Lectura8
 - Lectura9
- Tareas a realizar 2025 - 1
 - Planes para este sexto ciclo

The note content area is titled "Planes para este sexto ciclo". It contains a list of tasks:

- Clases regulares - Tareas próximas
 - Desarrollo de Software
 - Terminar las lecturas 1, 2 y 3
 - Actividad 1
 - Terminar lectura 4
 - Matemática computacional
 - Programación paralela
 - Administración de redes
 - Terminación de laboratorio 1
 - Decisión del proyecto final
- Acecom (2025) - 8 puntos bimestrales
 - Bimestre Enero - Febrero
 - Bimestre Marzo - Abril
 - Bimestre Mayo - Junio
 - Bimestre Julio - Agosto
- Proyecto Prometeo - Jetra
 - Investigación sobre SLAM con sensores de rango (LiDAR) y sensores inerciales (IMU)
 - Investigación de algoritmos SLAM incluyendo cámaras, GNSS o LiDAR
 - Investigación sobre Active SLAM y posibles relaciones con Deep Learning
- DataCamp - Obtener 10 000 exp mensuales
 - Marzo

Ejecutando el siguiente comando, veremos en muchísimo más detalle los commits realizados en el repositorio, mostrándose los cambios realizados:

```
$ git log -p
```



```
git log -p

commit 0e9f00f28bf909723e4fbfea7dbeeb2a3325469a (HEAD -> main, origin/main, origin/HEAD)
Author: bxcowo <a.flores.a@uni.pe>
Date: Tue Apr 1 16:37:33 2025 -0500

    vault backup: 2025-04-01 16:37:33

diff --git a/Tareas a realizar 2025 - 1/Planes para este sexto ciclo.md b/Tareas a realizar 2025 - 1/Planes para este sexto ciclo.md
index 05ff709..9e6dfel 100644
--- a/Tareas a realizar 2025 - 1/Planes para este sexto ciclo.md
+++ b/Tareas a realizar 2025 - 1/Planes para este sexto ciclo.md
@@ -1,15 +1,13 @@
- [ ] Clases regulares - Tareas próximas
  - [ ] Desarrollo de Software
    - [ ] Terminar las lecturas 1, 2 y 3
- - [ ] Actividad 1
+ - [x] Actividad 1
  - [ ] Terminar lectura 4
- - [ ] Matemática computacional
- - [ ] Programación paralela
- - [x] Exposición para el lunes
- - [ ] Proyecto final del curso (Reconocimiento facial)
- - [ ] Administración de redes
  - [x] Terminación de laboratorio 1
- - [x] Decisión de proyecto (Reconocimiento facial con redes neuronales)
+ - [ ] Decisión del proyecto final
- - [ ] Acecom (2025) - 8 puntos bimestrales
  - [x] Bimestre Enero - Febrero
- - [ ] Bimestre Marzo - Abril

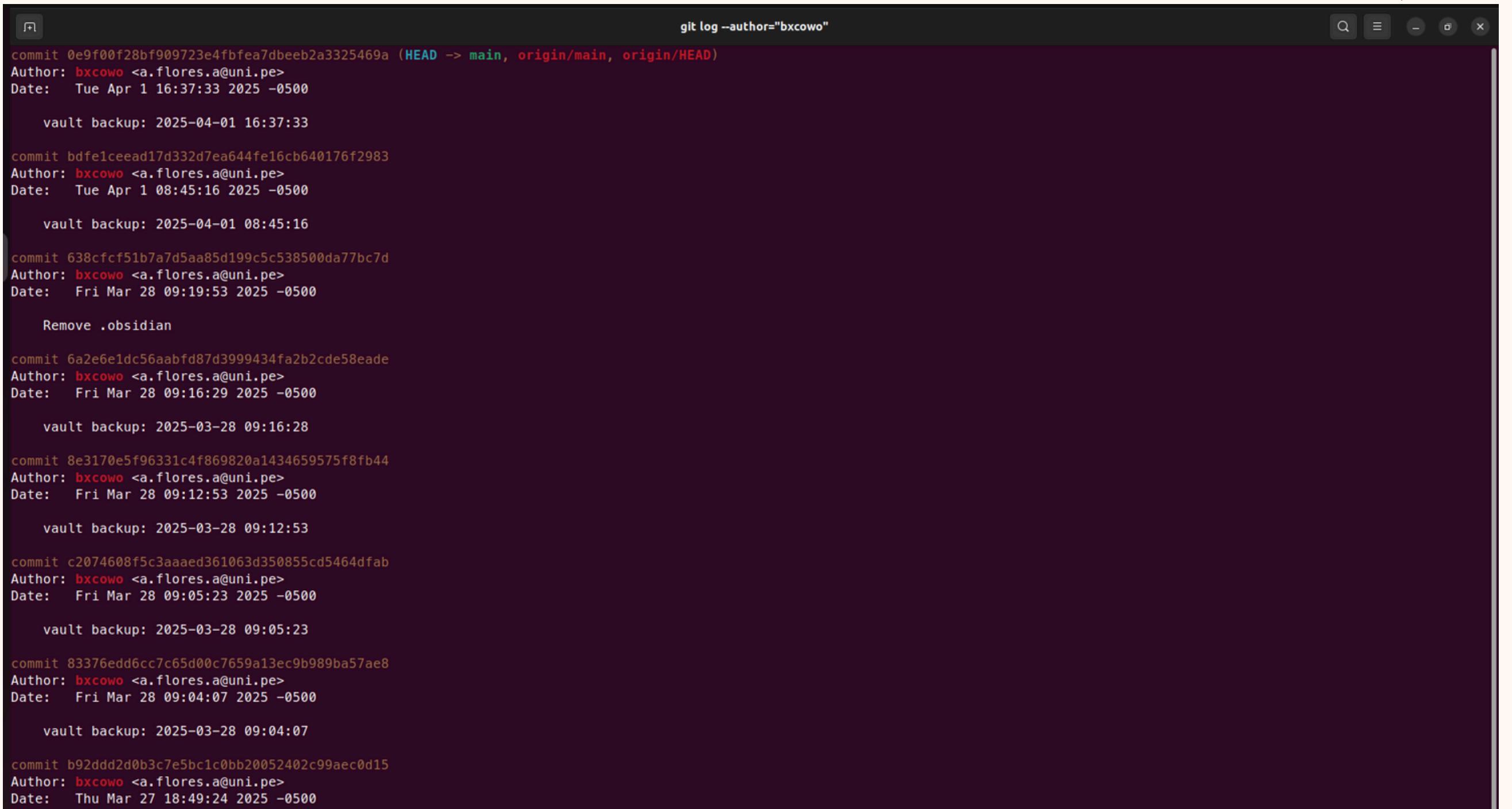
commit bdfe1ceead17d332d7ea644fe16cb640176f2983
Author: bxcowo <a.flores.a@uni.pe>
Date: Tue Apr 1 08:45:16 2025 -0500

    vault backup: 2025-04-01 08:45:16

diff --git a/Parallel programming/Final project.md b/Parallel programming/Final project.md
new file mode 100644
index 0000000..fd8593d
--- /dev/null
+++ b/Parallel programming/Final project.md
@@ -0,0 +1,30 @@
+## Tasks of the Selection Round of the Open Olympiad of St. Petersburg State University among Students and Young Specialists "Petropolitan Science (Re)Search" in the 2024/25 Academic Year in the Subject "Computational Technologies" for Master's/Specialist Students and Graduates
+
+## Task 1.
```

Dentro de trabajos colaborativos, uno puede contar con distintos participantes que puedan realizar aportaciones desde usuarios diferentes. De esto podemos realizar una filtración de commits apartir del nombre del autor mediante el siguiente comando:

```
$ git log --author="bxcowo"
```



```
git log --author="bxcowo"

commit 0e9f00f28bf909723e4fbfea7dbeeb2a3325469a (HEAD -> main, origin/main, origin/HEAD)
Author: bxcowo <a.flores.a@uni.pe>
Date: Tue Apr 1 16:37:33 2025 -0500

    vault backup: 2025-04-01 16:37:33

commit bdfe1ceead17d332d7ea644fe16cb640176f2983
Author: bxcowo <a.flores.a@uni.pe>
Date: Tue Apr 1 08:45:16 2025 -0500

    vault backup: 2025-04-01 08:45:16

commit 638cfccf51b7a7d5aa85d199c5c538500da77bc7d
Author: bxcowo <a.flores.a@uni.pe>
Date: Fri Mar 28 09:19:53 2025 -0500

    Remove .obsidian

commit 6a2e6e1dc56aabfd87d3999434fa2b2cde58eade
Author: bxcowo <a.flores.a@uni.pe>
Date: Fri Mar 28 09:16:29 2025 -0500

    vault backup: 2025-03-28 09:16:28

commit 8e3170e5f96331c4f869820a1434659575f8fb44
Author: bxcowo <a.flores.a@uni.pe>
Date: Fri Mar 28 09:12:53 2025 -0500

    vault backup: 2025-03-28 09:12:53

commit c2074608f5c3aaaed361063d350855cd5464dfab
Author: bxcowo <a.flores.a@uni.pe>
Date: Fri Mar 28 09:05:23 2025 -0500

    vault backup: 2025-03-28 09:05:23

commit 83376edd6cc7c65d00c7659a13ec9b989ba57ae8
Author: bxcowo <a.flores.a@uni.pe>
Date: Fri Mar 28 09:04:07 2025 -0500

    vault backup: 2025-03-28 09:04:07

commit b92ddd2d0b3c7e5bc1c0bb20052402c99aec0d15
Author: bxcowo <a.flores.a@uni.pe>
Date: Thu Mar 27 18:49:24 2025 -0500
```

Sin embargo, si sucediese algún error durante el desarrollo y se necesitase regresar a una versión anterior se pueden usar el siguiente comando que permita dicha reversión:

```
$ git revert HEAD
```

Se nos inicializará en la carpeta .git el archivo COMMIT_EDITMSG donde podremos un mensaje para el commit regresado.

```
GNU nano 6.2                               /media/bxco/obs:  
Revert "vault backup: 2025-04-01 16:37:33"  
  
This reverts commit 0e9f00f28bf909723e4fbfea7dbeeb2a3325469a.  
  
# Please enter the commit message for your changes. Lines starting  
# with '#' will be ignored, and an empty message aborts the commit.  
#  
# On branch main  
# Your branch is up to date with 'origin/main'.  
#  
# Changes to be committed:  
#       modified:   Tareas a realizar 2025 - 1/Planes para este sexto ciclo.md  
#
```

```
➜  /media/bxco/obsidian-git-sync ➜  main ➤ .....  
› git revert HEAD  
[main 4c25a47] Revert "vault backup: 2025-04-01 16:37:33"  
 1 file changed, 4 insertions(+), 2 deletions(-)
```

Ahora si verificasemos nuestro historial de commits usando \$ git log, notaremos los cambios hechos.

```
commit 4c25a4706b146d77a2d333ab91c3b9df1c3ff9e9 (HEAD -> main)  
Author: bxcowo <a.flores.a@uni.pe>  
Date:   Wed Apr 2 08:23:07 2025 -0500  
  
        Revert "vault backup: 2025-04-01 16:37:33"  
  
        This reverts commit 0e9f00f28bf909723e4fbfea7dbeeb2a3325469a.
```

Así mismo, podemos realizar un rebase interactivo que permita gestionar varios commits, para el siguiente ejemplo combinaremos 3 commits en uno solo, esto simplifica de gran forma los logs generados para una mejor gestión. Usaremos el siguiente comando:

```
$ git rebase -i HEAD~3
```

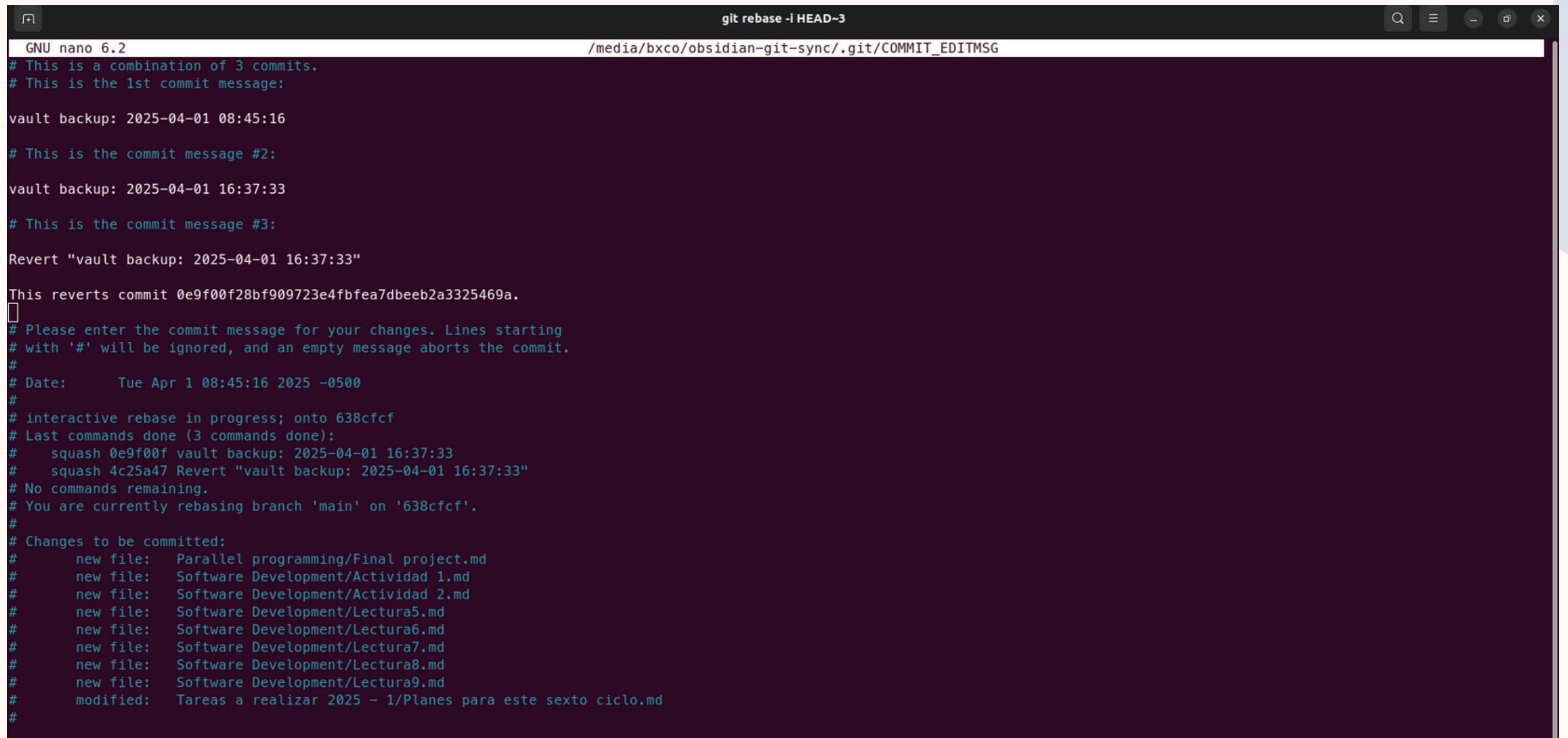
Es apartir de este comando que estamos considerando los tres últimos commits desde HEAD para alguno de los commandos presentes en los comentarios, usaremos squash para combinarlos todos en uno solo.



```
GNU nano 6.2                               /media/bxco/obsidian-git-sync/.git/rebase-merge/git-rebase-todo *
pick bdfe1ce vault backup: 2025-04-01 08:45:16
squash 0e9f00f vault backup: 2025-04-01 16:37:33
squash 4c25a47 Revert "vault backup: 2025-04-01 16:37:33"

# Rebase 638cfcc..4c25a47 onto 638cfcc (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                               commit's log message, unless -C is used, in which case
#                               keep only this commit's message; -c is same as -C but
#                               opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
#           create a merge commit using the original merge commit's
#           message (or the oneline, if no original merge commit was
#           specified); use -c <commit> to reword the commit message
# u, update-ref <ref> = track a placeholder for the <ref> to be updated
#                     to this position in the new commits. The <ref> is
#                     updated at the end of the rebase
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
```

Al haber realizado correctamente la ejecución del comando, se nos abrirá nuevamente el editor para la implementación de un mensaje asociado a ellos.



```
git rebase -i HEAD~3
GNU nano 6.2
/media/bxco/obsidian-git-sync/.git/COMMIT_EDITMSG

# This is a combination of 3 commits.
# This is the 1st commit message:

vault backup: 2025-04-01 08:45:16

# This is the commit message #2:

vault backup: 2025-04-01 16:37:33

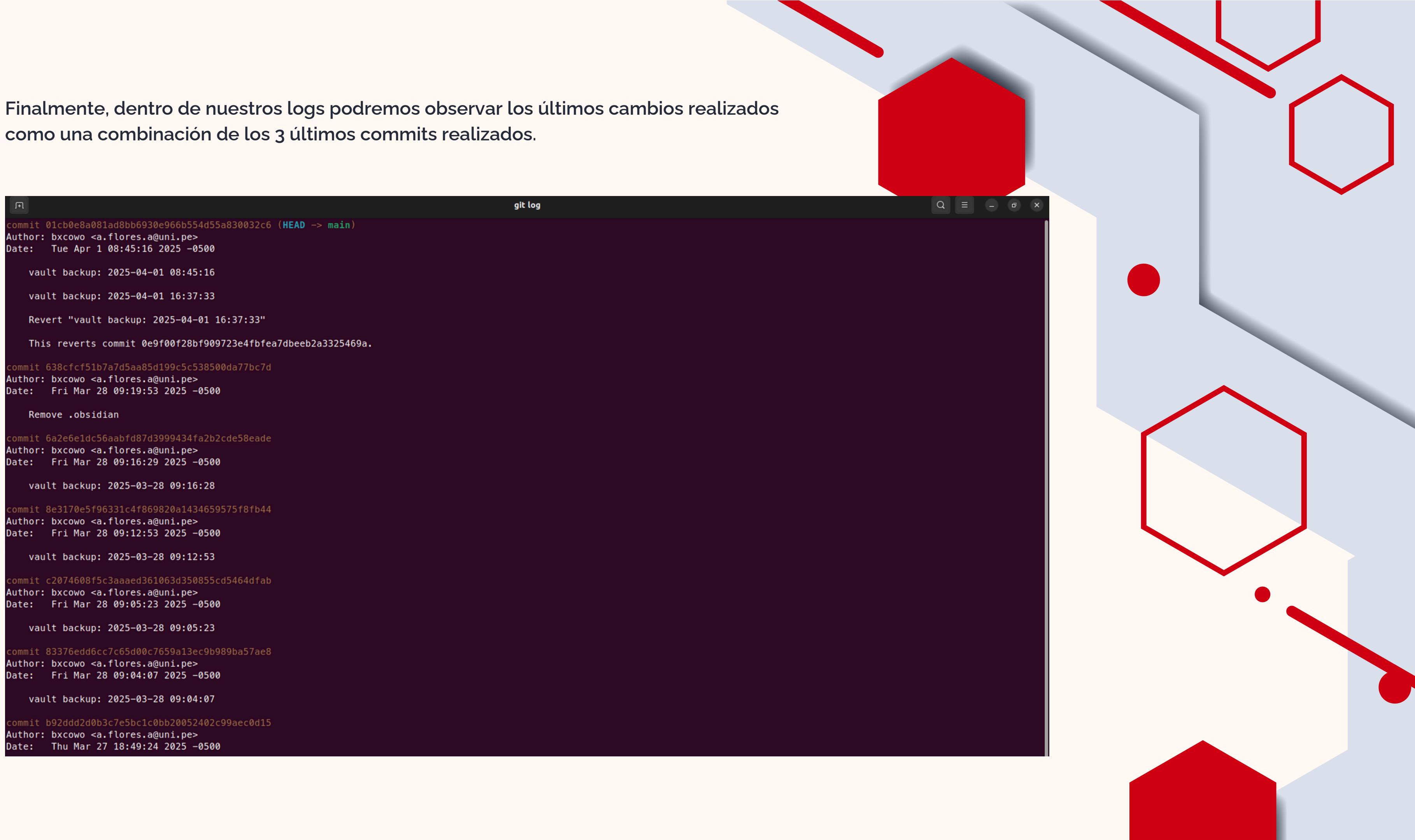
# This is the commit message #3:

Revert "vault backup: 2025-04-01 16:37:33"

This reverts commit 0e9f00f28bf909723e4fbfea7dbeeb2a3325469a.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Tue Apr 1 08:45:16 2025 -0500
#
# interactive rebase in progress; onto 638cfcc
# Last commands done (3 commands done):
#   squash 0e9f00f vault backup: 2025-04-01 16:37:33
#   squash 4c25a47 Revert "vault backup: 2025-04-01 16:37:33"
# No commands remaining.
# You are currently rebasing branch 'main' on '638cfcc'.
#
# Changes to be committed:
#   new file:  Parallel programming/Final project.md
#   new file:  Software Development/Actividad 1.md
#   new file:  Software Development/Actividad 2.md
#   new file:  Software Development/Lectura5.md
#   new file:  Software Development/Lectura6.md
#   new file:  Software Development/Lectura7.md
#   new file:  Software Development/Lectura8.md
#   new file:  Software Development/Lectura9.md
#   modified: Tareas a realizar 2025 - 1/Planes para este sexto ciclo.md
```

Finalmente, dentro de nuestros logs podremos observar los últimos cambios realizados como una combinación de los 3 últimos commits realizados.



```
git log
```

```
commit 01cb0e8a081ad8bb6930e966b554d55a830032c6 (HEAD -> main)
Author: bxcowo <a.flores.a@uni.pe>
Date:   Tue Apr 1 08:45:16 2025 -0500

    vault backup: 2025-04-01 08:45:16
    vault backup: 2025-04-01 16:37:33
    Revert "vault backup: 2025-04-01 16:37:33"
    This reverts commit 0e9f00f28bf909723e4fbfea7dbeeb2a3325469a.

commit 638cfccf51b7a7d5aa85d199c5c538500da77bc7d
Author: bxcowo <a.flores.a@uni.pe>
Date:   Fri Mar 28 09:19:53 2025 -0500

    Remove .obsidian

commit 6a2e6e1dc56aabfd87d3999434fa2b2cde58eade
Author: bxcowo <a.flores.a@uni.pe>
Date:   Fri Mar 28 09:16:29 2025 -0500

    vault backup: 2025-03-28 09:16:28

commit 8e3170e5f96331c4f869820a1434659575f8fb44
Author: bxcowo <a.flores.a@uni.pe>
Date:   Fri Mar 28 09:12:53 2025 -0500

    vault backup: 2025-03-28 09:12:53

commit c2074608f5c3aaaed361063d350855cd5464dfab
Author: bxcowo <a.flores.a@uni.pe>
Date:   Fri Mar 28 09:05:23 2025 -0500

    vault backup: 2025-03-28 09:05:23

commit 83376edd6cc7c65d00c7659a13ec9b989ba57ae8
Author: bxcowo <a.flores.a@uni.pe>
Date:   Fri Mar 28 09:04:07 2025 -0500

    vault backup: 2025-03-28 09:04:07

commit b92ddd2d0b3c7e5bc1c0bb20052402c99aec0d15
Author: bxcowo <a.flores.a@uni.pe>
Date:   Thu Mar 27 18:49:24 2025 -0500
```

Otra forma alternativa de visualización de commits será a través de una representación gráfica que permita un entendimiento mucho más adecuado del log asociado al repositorio. Para realizarlo ejecutaremos el siguiente comando:

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

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

* 01cb0e8 (HEAD -> main) vault backup: 2025-04-01 08:45:16
| * 0e9f00f (origin/main, origin/HEAD) vault backup: 2025-04-01 16:37:33
| * bdfe1ce vault backup: 2025-04-01 08:45:16
|/
* 638cfcc Remove .obsidian
* 6a2e6e1 vault backup: 2025-03-28 09:16:28
* 8e3170e vault backup: 2025-03-28 09:12:53
* c207460 vault backup: 2025-03-28 09:05:23
* 83376ed vault backup: 2025-03-28 09:04:07
* b92ddd2 Fix the merge
* ff54768 Merge branch 'main' of https://github.com/bxcowo/obsidian-git-sync
|\
| * 115d703 Create README.md
* | 3b4eaal Add Laras lectures
|/
* c4030bd Add all left files of my vault
* 1f21df7 first commit
* 8a9bcb9 first commit
(END)
```

Notamos que los commits son representados como asteriscos y sus cambios realizados son formados como líneas que las relacionan

EJERCICIO 3



OBJETIVO

Practicar la creación de ramas desde commits específicos y comprender cómo Git maneja las referencias históricas.

Crear una nueva rama desde un commit específico:

- Usa el historial de commits (`git log --oneline`) para identificar un commit antiguo desde el cual crear una nueva rama:

```
$ git log --oneline
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git log --oneline
fe34bde (HEAD -> master) Resolve merge conflict between master and feature/advanced-feature
01d7d20 Update main.py message in master brach
fe87c76 Add greet function in advanced feature
4f4a679 Add greet function in advanced feature
19474a5 (feature/another-new-feature, develop) add main.py
7c10371 Set up the repository base documentation
cf05b8d Initial commit with README.md
```

- Crea una nueva rama `bugfix/rollback-feature` desde el commit donde se añadio el `main.py`:

```
$ git branch bugfix/rollback-feature 19474a5
```

```
$ git checkout bugfix/rollback-feature
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git branch bugfix/rollback-feature 19474a5
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git checkout bugfix/rollback-feature
Switched to branch 'bugfix/rollback-feature'
```

Modificar y confirmar cambios en la nueva rama:

- Realiza algunas modificaciones en `main.py` que simulen una corrección de errores:

```
def greet():
    print('Fixed bug in feature')
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kap...  main.py *
GNU nano 6.2
print('Hello World')
def greet():
    print('Fixed bug in feature')

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

- Añade y confirma los cambios en la nueva rama:

```
$ git add main.py
```

```
$ git commit -m "Fix bug in rollback feature"
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git add main.py
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git commit -m "Fix bug in rollback feature"
[bugfix/rollback-feature 1160b06] Fix bug in rollback feature
 1 file changed, 2 insertions(+)
```

Fusionar los cambios en la rama principal:

- Cambia de nuevo a la rama main y fusiona la rama bugfix/rollback-feature:

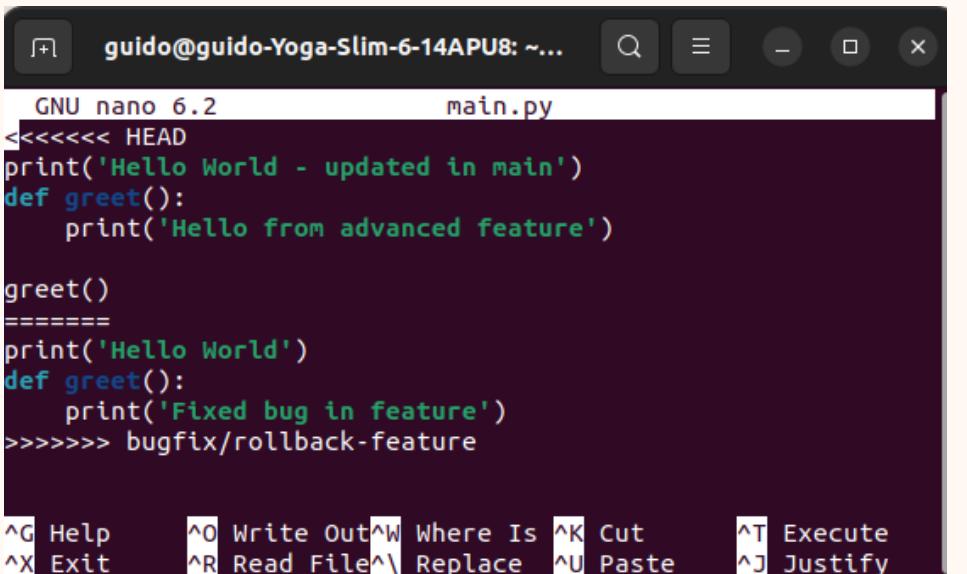
```
$ git checkout main
```

```
$ git merge bugfix/rollback-feature
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git checkout master
Switched to branch 'master'
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git merge bugfix/rollback-feature
Auto-merging main.py
CONFLICT (content): Merge conflict in main.py
Automatic merge failed; fix conflicts and then commit the result.
```

- Arreglar conflictos manualmente

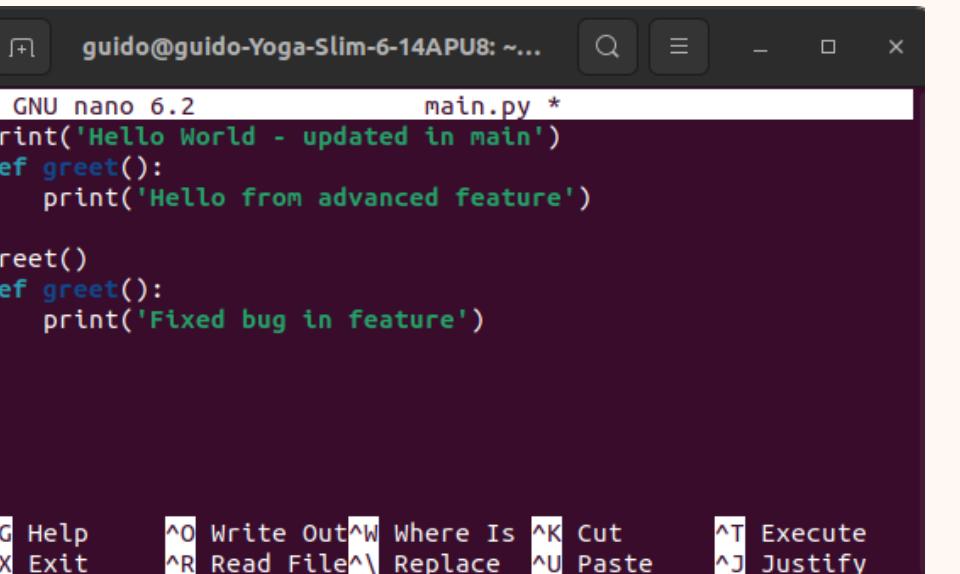
```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ nano main.py
```



```
GNU nano 6.2          main.py
<<<< HEAD
print('Hello World - updated in main')
def greet():
    print('Hello from advanced feature')

greet()
=====
print('Hello World')
def greet():
    print('Fixed bug in feature')
>>>> bugfix/rollback-feature

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



```
GNU nano 6.2          main.py *
print('Hello World - updated in main')
def greet():
    print('Hello from advanced feature')

greet()
def greet():
    print('Fixed bug in feature')

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

- Añade el archivo resuelto y completamos el merge

```
$ git add main.py
```

```
$ git commit -m "resolve marge and fix bug in rollback feature"
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git add main.py
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git commit -m "resolve marge and fix bug in rollback feature"
[master af57515] resolve marge and fix bug in rollback feature
```

Explorar el historial después de la fusión:

- Usa `git log` y `git log --graph` para ver cómo se ha integrado el commit en el historial:

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

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git log --graph --oneline
*   af57515 (HEAD -> master) resolve marge and fix bug in rollback feature
|\ \
| * 1160b06 (bugfix/rollback-feature) 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
| |
* 19474a5 (feature/another-new-feature, develop) add main.py
* 7c10371 Set up the repository base documentation
* cf05b8d Initial commit with README.md
```

Eliminar la rama bugfix/rollback-feature:

- Una vez fusionados los cambios, elimina la rama `bugfix/rollback-feature`:

```
$ git branch -d bugfix/rollback-feature
```

```
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git branch -d bugfix/rollback-feature
Deleted branch bugfix/rollback-feature (was 1160b06).
guido@guido-Yoga-Slim-6-14APU8:~/Desktop/kapumota-repo$ git branch
  develop
  feature/another-new-feature
  feature/login
  hotfix/bugfix
* master
```

EJERCICIO 4



OBJETIVO

Comprender cómo usar `git reset` y `git restore` para deshacer cambios en el historial y en el área de trabajo

Se hará nuevamente uso del repositorio en Obsidian para la realización del siguiente ejercicio. Comenzaremos realizando modificaciones en unos de los archivos definidos en dicho repositorio.

The screenshot shows the Obsidian interface with the 'Introduction to Optimization' note selected in the sidebar. The main content area displays the following text:

Introduction to Optimization

1. The Concept of Optimization

Optimization is a fundamental concept that permeates many aspects of our world. At its core, optimization is concerned with finding the "best" solution to a problem. The concept of "best" is quantified using an objective function, which assigns a numerical value to each possible solution. The mathematical framework of optimization offers a systematic approach to decision-making in complex scenarios. Whether we're maximizing profit in a business context, minimizing weight in an engineering design, or finding the shortest path in a network, optimization provides the tools to make informed decisions.

The screenshot shows the Obsidian interface with the 'Introduction to Optimization' note selected in the sidebar. The main content area displays the following text:

Introduction to Optimization

1. The Concept of Optimization

Optimization is the mathematical practice of finding the best suited values according to a declared problem with its corresponding objective function. Their applications go from engineering to economics, often called as the best approach to solve different real world problems.

2. The Basic Optimization Problem

2.1 Mathematical Formulation

The standard form of an optimization problem is:

$$\text{minimize } f(x)$$

Where:

- x represents the decision variables (also called design variables)

Dicho cambio deberá de ser subido al staging area, para esto realizaremos uso de:
\$ git add Computational\ Mathematics/Introduction\ to\ Optimization.md

```
q ➔ /media/bxco/obsidian-git-sync ➔ 🐈 main !1 .....  
❯ git add Computational\ Mathematics/Introduction\ to\ Optimization.md
```

Posterior a dicho proceso, ahora realizaremos un commit con dichos cambios para guardarlos dentro de nuestra base de datos local. Ejecutamos el siguiente comando:

```
$ git commit -m 'Edit Introduction to Optimization.md'
```

```
q ➔ /media/bxco/obsidian-git-sync ➔ 🐈 main +1 ....  
❯ git commit -m 'Edit Introduction to Optimization.md'  
[main bfd460b] Edit Introduction to Optimization.md  
1 file changed, 1 insertion(+), 4 deletions(-)
```

Podemos verificar la realización de dicho commit si verificamos nuestro historial de commits ejecutando:

```
$ git log
```

```
commit bfd460b4c10de7f66da8998de0c1c4b019c11174 (HEAD -> main)
Author: bxcowo <a.flores.a@uni.pe>
Date:   Fri Apr 4 12:39:30 2025 -0500

Edit Introduction to Optimization.md
```

Ahora si se nos presentase el caso que deseemos revertir nuestro cambio registrado, deberemos de ejecutar el siguiente comando:

```
$ git reset --hard HEAD~1
```

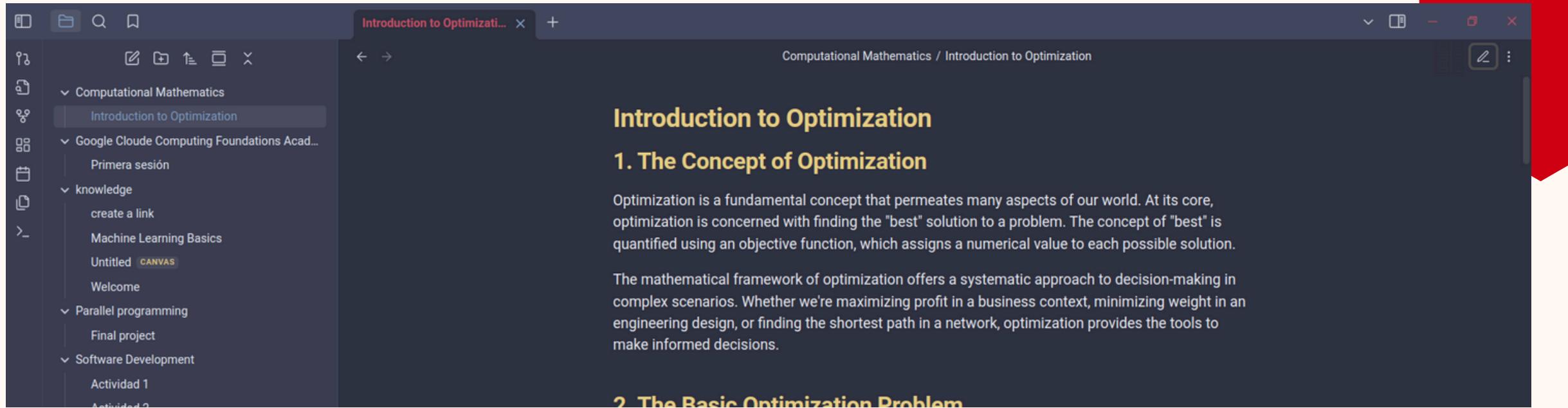
```
❯ git reset --hard HEAD~1
HEAD is now at f03df8a vault backup: 2025-04-04 12:22:46
```

Con lo realizado podremos verificar dentro de nuestros archivo y dentro de nuestro log que el último commit fue cambiado a un anterior al que se realizó.

```
commit f03df8af646ee080eab9704074cf2379aa5a5909 (HEAD -> main, origin/main, origin/HEAD)
Author: bxcowo <a.flores.a@uni.pe>
Date:   Fri Apr 4 12:22:46 2025 -0500

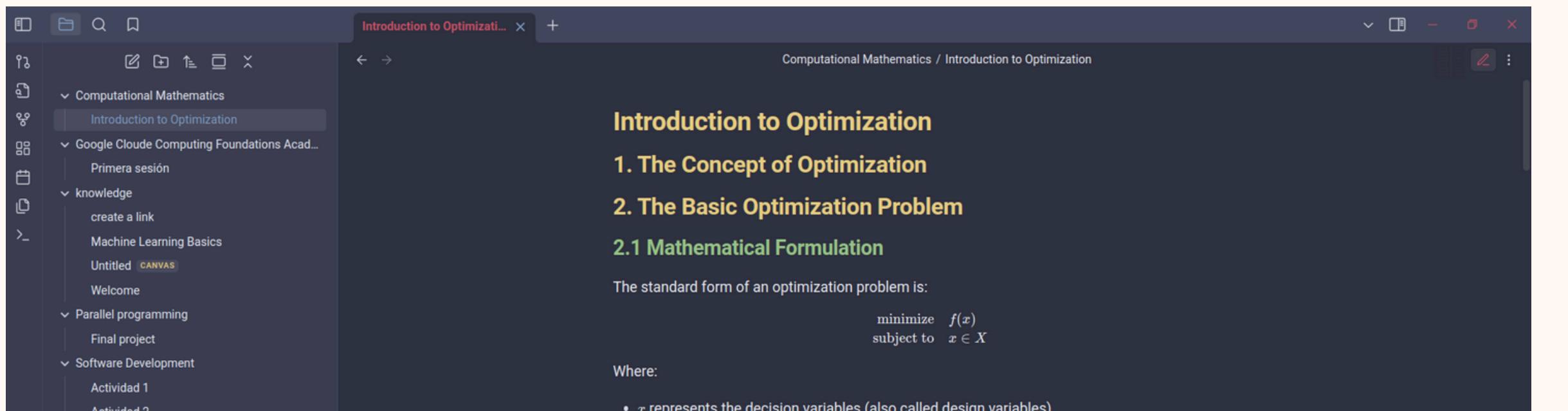
    vault backup: 2025-04-04 12:22:46
```

Se puede verificar dentro de los archivos de obsidian que regresamos a nuestra versión original.



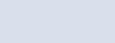
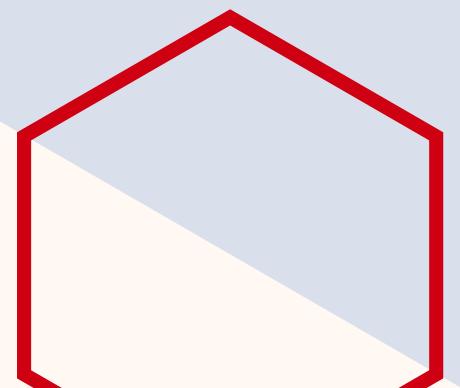
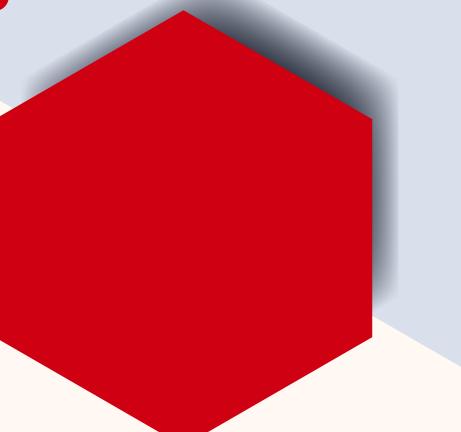
Ahora bien, se pueden revertir cambios registrados en nuestro repositorio local, pero estos procedimientos también pueden ser aplicados para aquellos que sean no registrados.

Veamos nuevamente una modificación del mismo archivo, donde accidentalmente borramos la definición inicial.



Ejecutaremos el siguiente comando para asegurarnos que nuestros cambios aún no hayan sido registrados:

```
$ git status
```



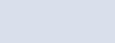
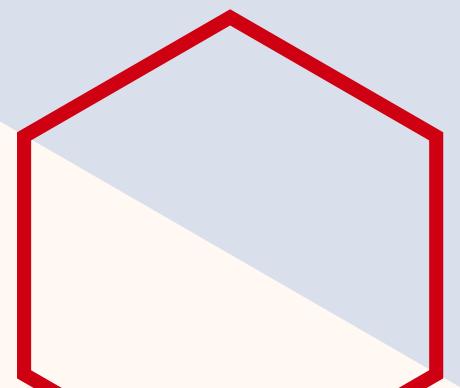
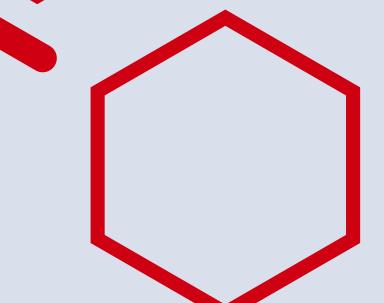
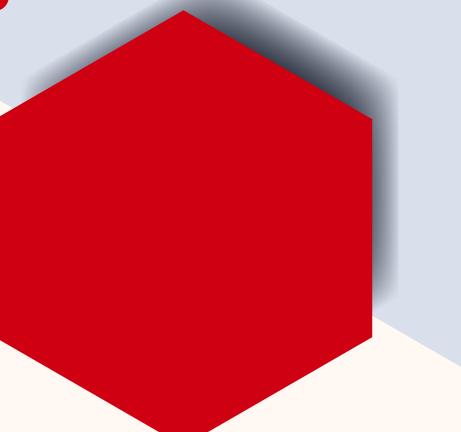
```
q ➔ /media/bxco/obsidian-git-sync ➔ 🐫 main !1
> git status
On branch main
Your branch is up to date with 'origin/main'.

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:   Computational Mathematics/Introduction to Optimization.md

no changes added to commit (use "git add" and/or "git commit -a")
```

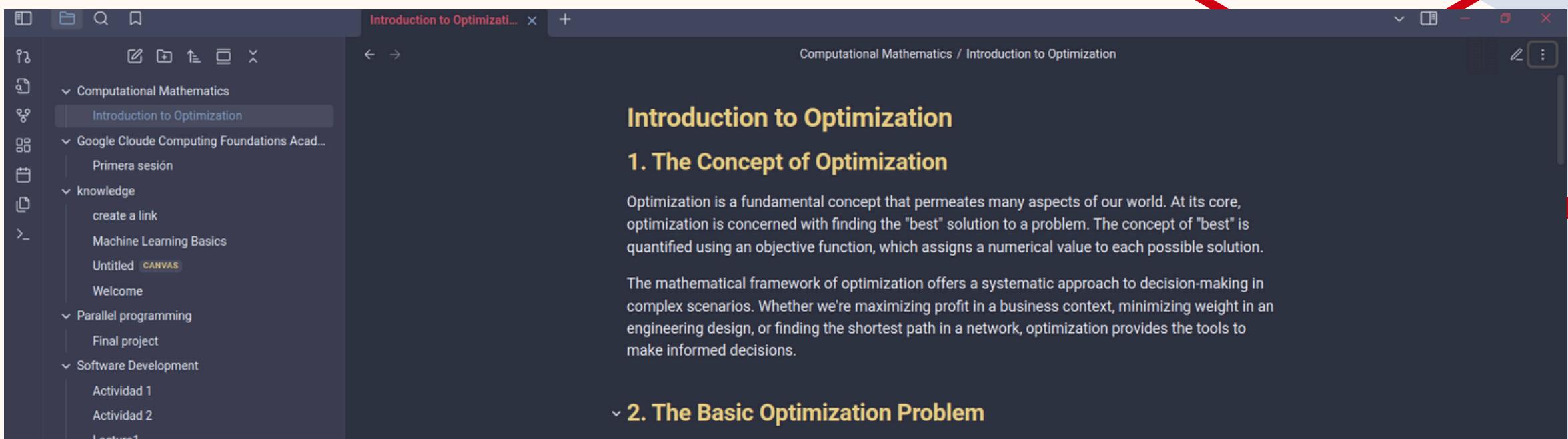
Para revertir dichos cambios sin necesariamente tener que utilizar 'Control + z', podemos utilizar el siguiente comando que especifique restaurar los cambios hechos en dicho archivo:

```
$ git restore Computational\ Mathematics/Introduction\ to\ Optimization.md
```



```
q ➔ /media/bxco/obsidian-git-sync ➔ 🐫 main !1
> git restore Computational\ Mathematics/Introduction\ to\ Optimization.md
```

Verificamos dentro de nuestros archivos que regresamos con nuestras definiciones previamente eliminadas.



EJERCICIO 5



OBJETIVO

Simular un flujo de trabajo colaborativo utilizando ramas y pull requests.

Crear un nuevo repositorio remoto:

- Usa GitHub o GitLab para crear un nuevo repositorio remoto y clónalo localmente: `$ git clone <URL-del-repositorio>`

```
diegodev@HPavilion:~/Desktop/dev-practice$ git clone git@github.com:Ox-Chema-x0/team-practice.git
Cloning into 'team-practice'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
diegodev@HPavilion:~/Desktop/dev-practice$
```

Crear una nueva rama para desarrollo de una característica:

- En tu repositorio local, crea una nueva rama feature/team-feature:
`$ git branch feature/team-feature`
`$ git checkout feature/team-feature`

```
diegodev@HPavilion:~/Desktop/dev-practice$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
diegodev@HPavilion:~/Desktop/dev-practice$ git branch feature/team-feature
diegodev@HPavilion:~/Desktop/dev-practice$ git checkout feature/team-feature
Switched to branch 'feature/team-feature'
```

Realizar cambios y enviar la rama al repositorio remoto:

- Realiza cambios en los archivos del proyecto y confírmalos:

```
$ echo "print('Collaboration is key!') > collaboration.py
```

```
$ git add .
```

```
$ git commit -m "Add collaboration script"
```

```
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$ echo "print('Collaboration is key!') > collaboration.py"
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$ git add .
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$ git commit -m "Add collaboration script"
[feature/team-feature 4cbad76] Add collaboration script
 1 file changed, 1 insertion(+)
  create mode 100644 collaboration.py
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$
```

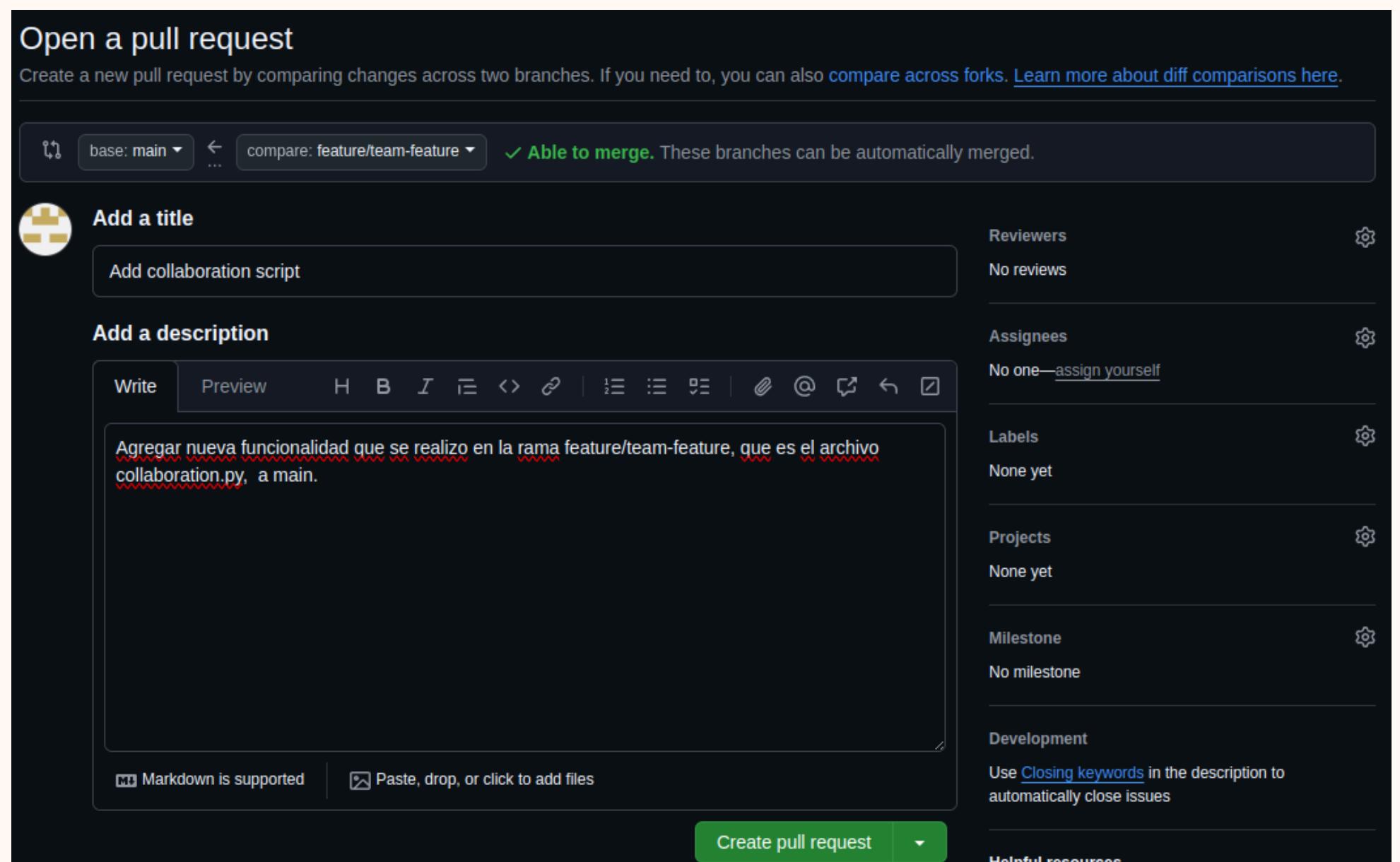
- Envía la rama al repositorio remoto:

```
$ git push origin feature/team-feature
```

```
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$ git push origin feature/team-feature
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), 325 bytes | 325.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/team-feature' on GitHub by visiting:
remote:     https://github.com/0x-Chema-x0/team-practice/pull/new/feature/team-feature
remote:
To github.com:0x-Chema-x0/team-practice.git
 * [new branch]      feature/team-feature -> feature/team-feature
```

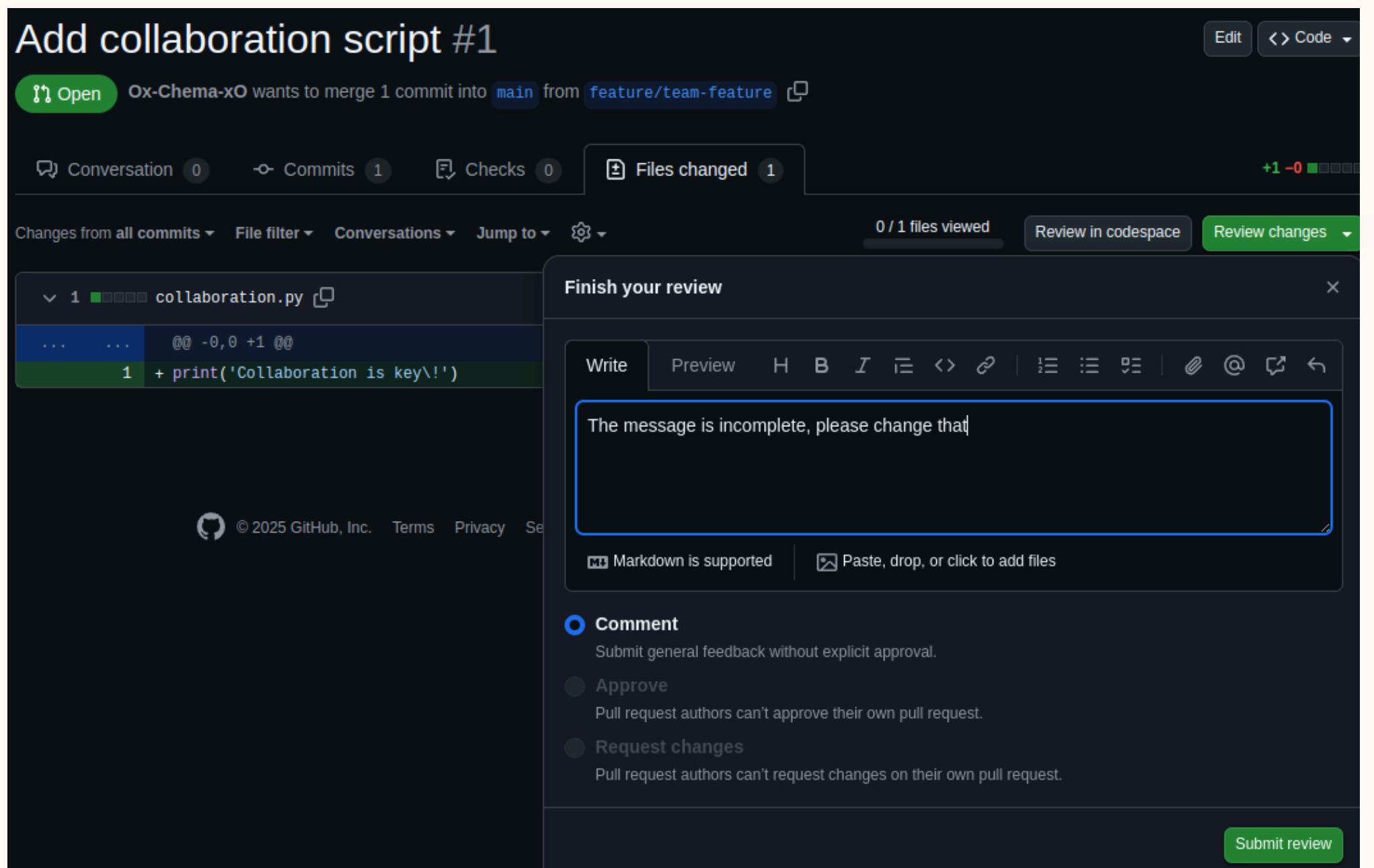
Abrir un Pull Request:

- Abre un Pull Request (PR) en la plataforma remota (GitHub/GitLab) para fusionar feature/team-feature con la rama main.
- Añade una descripción detallada del PR, explicando los cambios realizados y su propósito.



Revisar y fusionar el Pull Request:

- Simula la revisión de código, comenta en el PR y realiza cualquier cambio necesario basado en la retroalimentación.



Revisar y fusionar el Pull Request:

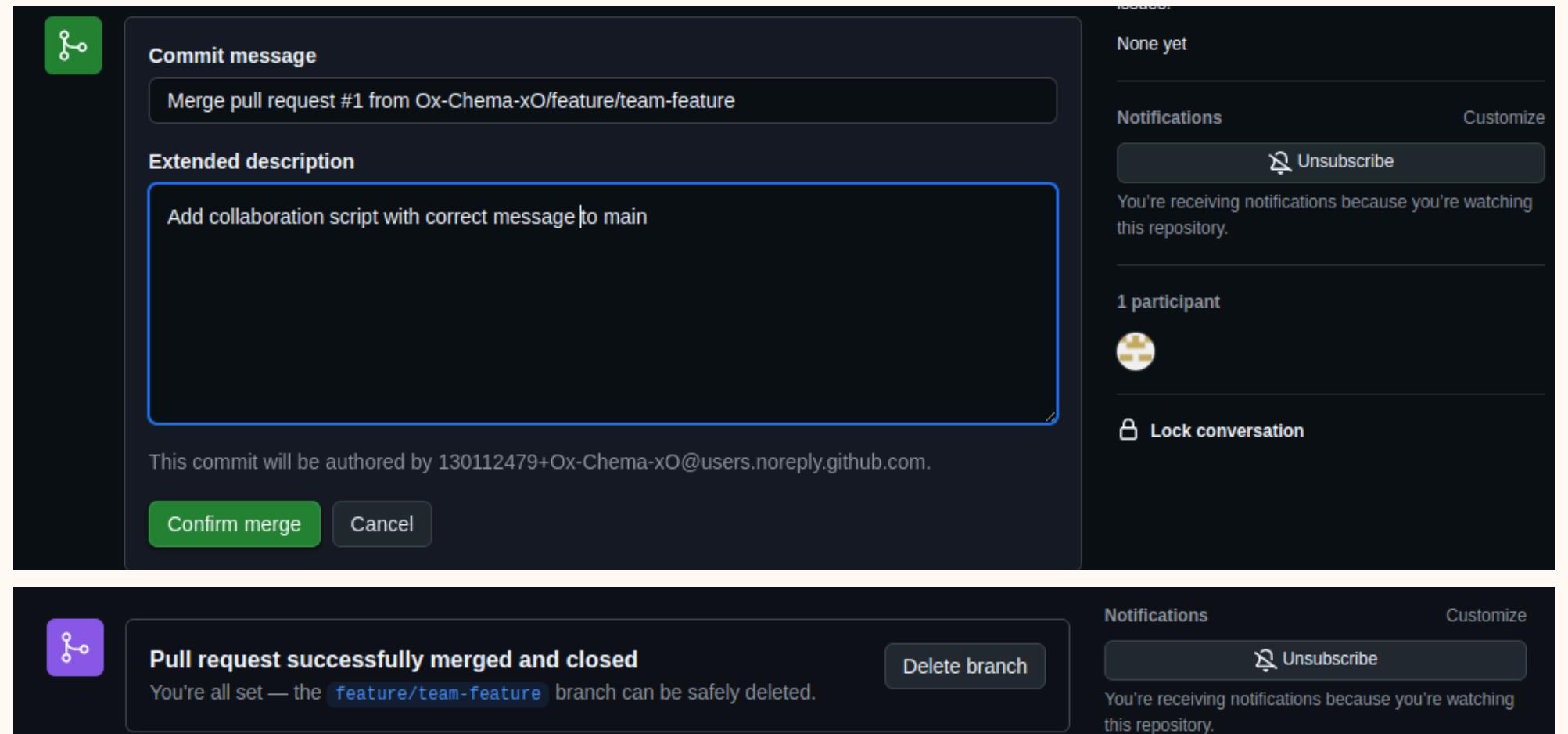
- Simula la revisión de código, comenta en el PR y realiza cualquier cambio necesario basado en la retroalimentación.

```
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$ git branch
* feature/team-feature
  main
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$ git status
On branch feature/team-feature
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:   collaboration.py

no changes added to commit (use "git add" and/or "git commit -a")
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$ git add collaboration.py
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$ git commit -m "Mensaje correcto para collaboration file"
[feature/team-feature fbee8a] Mensaje correcto para collaboration file
 1 file changed, 1 insertion(+), 1 deletion(-)
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$ git push origin feature/team-feature
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 372 bytes | 372.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:0x-Chema-x0/team-practice.git
  4cbad76..fbee8a  feature/team-feature -> feature/team-feature
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$
```

Revisar y fusionar el Pull Request:

- Una vez aprobado, fusiona el PR en la rama main.



Eliminar la rama remota y local:

- Despues de la fusión, elimina la rama tanto local como remotamente:

```
$ git branch -d feature/team-feature
```

```
$ git push origin --delete feature/team-feature
```

```
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$ git branch -d feature/team-feature
Deleted branch feature/team-feature (was fbeea8a).
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$ git push origin --delete feature/team-feature
To github.com:Ox-Chema-xO/team-practice.git
 - [deleted]      feature/team-feature
diegodev@HPavilion:~/Desktop/dev-practice/team-practice$
```

EJERCICIO 6



OBJETIVO

Aprender a aplicar commits específicos a otra rama utilizando git cherry-pick y a guardar temporalmente cambios no confirmados utilizando git stash.

Hacer cambios en main.py y confirmarlos:

- Realiza y confirma varios cambios en main.py en la rama main:

```
$ echo "print('Cherry pick this!')" >> main.py  
$ git add main.py  
$ git commit -m "Add cherry-pick example"
```

```
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ echo "print('Cherry pick this!')" >> main.py  
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git add main.py  
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git commit -m "Add cherry-pick example"  
[main ef2d04d] Add cherry-pick example  
 1 file changed, 1 insertion(+)
```

Crear una nueva rama y aplicar el commit específico:

- Crea una nueva rama feature/cherry-pick y aplícale el commit específico:

```
$ git branch feature/cherry-pick  
$ git checkout feature/cherry-pick
```

```
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git branch feature/cherry-pick  
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git checkout feature/cherry-pick  
Switched to branch 'feature/cherry-pick'
```

Crear una nueva rama y aplicar el commit específico:

- Crea una nueva rama feature/cherry-pick y aplícale el commit específico:

```
$ git log  
$ git cherry-pick <commit-hash>
```

```
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git log  
commit ef2d04d85243b4a1b695351f3a7142aa06277a49 (HEAD -> main)  
Author: Diego <manuelchema999@gmail.com>  
Date:   Fri Apr 4 09:32:40 2025 -0500  
  
    Add cherry-pick example  
  
commit 49b487def6559570da8d2354fd46d9c461bdeeca (feature/cherry-pick)  
Author: Diego <manuelchema999@gmail.com>  
Date:   Thu Apr 3 08:35:44 2025 -0500  
  
    add main.py  
  
commit 444795fad309ff90d18b5783a2e84dc0baf75910  
Author: Diego <manuelchema999@gmail.com>  
Date:   Thu Apr 3 08:32:43 2025 -0500  
  
    Initial commit with README.md
```

```
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git switch feature/cherry-pick  
Switched to branch 'feature/cherry-pick'  
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git cherry-pick ef2d04d85243b4a1b695351f3a71  
42aa06277a49  
[feature/cherry-pick b14aae1] Add cherry-pick example  
Date: Fri Apr 4 09:32:40 2025 -0500  
1 file changed, 1 insertion(+)
```

Guardar temporalmente cambios no confirmados:

- Realiza algunos cambios en main.py pero no los confirme:

```
$ echo "This change is stashed" >> main.py
```

```
$ git status
```

```
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ echo "This change is stashed" >> main.py
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git status
On branch main
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")
```

- Guarda temporalmente estos cambios utilizando git stash:

```
$ git stash
```

```
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git stash
Saved working directory and index state WIP on main: ef2d04d Add cherry-pick example
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git status
On branch main
nothing to commit, working tree clean
```

Aplicar los cambios guardados:

- Realiza otros cambios y confírmalos si es necesario.
- Luego, recupera los cambios guardados anteriormente:

```
$ git stash pop
```

Aplicar los cambios guardados:

- Realiza otros cambios y confírmalos si es necesario.
- Luego, recupera los cambios guardados anteriormente:

```
$ git stash pop
```

```
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ echo "new change for practice that" >> main.py
```

```
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git add main.py  
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git commit -m "Add new change into main.py"  
[main 8078ca4] Add new change into main.py  
 1 file changed, 1 insertion(+)
```

- Luego, recupera los cambios guardados anteriormente:

```
$ git stash pop
```

```
diegodev@HPavilion:~/Desktop/dev-practice/activ4-git$ git stash pop  
Auto-merging main.py  
CONFLICT (content): Merge conflict in main.py  
On branch main  
Unmerged paths:  
  (use "git restore --staged <file>..." to unstage)  
    (use "git add <file>..." to mark resolution)  
      both modified: main.py  
  
no changes added to commit (use "git add" and/or "git commit -a")  
The stash entry is kept in case you need it again.
```

Revisar el historial y confirmar la correcta aplicación de los cambios:

- Usa git log para revisar el historial de commits y verificar que todos los cambios se han aplicado correctamente.

```
commit ba38e4d26d489c22a6db3a2acdb0f38b04393e17 (HEAD -> main)
Author: Diego <manuelchema999@gmail.com>
Date:   Fri Apr 4 10:35:48 2025 -0500

    Resolve conflicts after git stash pop

commit 8078ca4002e8afdf56d5f82148b34fae89c6979ff
Author: Diego <manuelchema999@gmail.com>
Date:   Fri Apr 4 10:26:44 2025 -0500

    Add new change into main.py

commit ef2d04d85243b4a1b695351f3a7142aa06277a49
Author: Diego <manuelchema999@gmail.com>
Date:   Fri Apr 4 09:32:40 2025 -0500

    Add cherry-pick example

commit 49b487def6559570da8d2354fd46d9c461bdeeca
Author: Diego <manuelchema999@gmail.com>
Date:   Thu Apr 3 08:35:44 2025 -0500

    add main.py
```



**MUCHAS
GRACIAS**