

DESARROLLO DE SOFTWARE

ACTIVIDAD 4

Aarón Flores Alberca



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.

Considerando un repositorio nuevo vacío creamos una nueva rama llamada feature/advanced-feature desde la rama main:

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

Añadimos en dicha rama un archivo main.py con el siguiente contenido.

```
$ vim main.py
```

Posterior a ello commiteamos dichos cambios con los siguientes comandos:

```
$ git add main.py  
$ git commit -m 'Add greet function in advanced feature'
```

```
bxco@bxco-NBD-WXX9:~/pregunta1$ git branch feature/advanced-feature  
bxco@bxco-NBD-WXX9:~/pregunta1$ git checkout feature/advanced-feature  
Cambiado a rama 'feature/advanced-feature'  
bxco@bxco-NBD-WXX9:~/pregunta1$ git branch  
* feature/advanced-feature  
  main
```

```
def greet():  
    print("Hola desde feature/advanced-feature")  
  
greet()  
~  
~  
~  
~  
~  
~  
~
```

```
bxco@bxco-NBD-WXX9:~/pregunta1$ git add main.py  
bxco@bxco-NBD-WXX9:~/pregunta1$ git commit -m 'Add greet function in advanced feature'  
[feature/advanced-feature 2e56f53] Add greet function in advanced feature  
 1 file changed, 4 insertions(+)  
create mode 100644 main.py
```

Regresemos a la rama main mediante el siguiente comando:

```
$ git checkout main
```

Dentro de este intentemos crear un nuevo archivo main.py que tenga un contenido diferente, usemos nuevamente

```
$ vim main.py
```

Luego confirmamos estos cambios dentro de la rama main

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

```
$ git commit -m 'Add main.py in main branch'
```

Ahora intentemos combinar ambas ramas en una sola usando:

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

Cuando ejecutemos dicho comando, obtendremos un error de conflictos en dicha acción que deberemos de resolver manualmente:

```
$ vim main.py
```

La solución consistirá en remover los símbolos de las ramas que se desean combinar y ejecutar nuevamente un commit.

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

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

```
bxco@bxco-NBD-WXX9:~/pregunta1$ git checkout main
Cambiado a rama 'main'
```

```
def greet():
    print("Hola desde la rama main")

greet()
~
~
~
~
```

```
bxco@bxco-NBD-WXX9:~/pregunta1$ git add main.py
bxco@bxco-NBD-WXX9:~/pregunta1$ git commit -m 'Add main.py in main branch'
[main b23c1cc] Add main.py in main branch
 1 file changed, 4 insertions(+)
 create mode 100644 main.py
```

```
bxco@bxco-NBD-WXX9:~/pregunta1$ git merge feature/advanced-feature
Auto-fusionando main.py
CONFLICTO (agregar/agregar): Conflicto de fusión en main.py
Fusión automática falló; arregle los conflictos y luego realice un commit con el resultado.
```

```
def greet():
<<<<< HEAD
    print("Hola desde la rama main")
=====
    print("Hola desde feature/advanced-feature")
>>>> feature/advanced-feature

greet()
~
~
```

```
bxco@bxco-NBD-WXX9:~/pregunta1$ git add main.py
bxco@bxco-NBD-WXX9:~/pregunta1$ git commit -m 'Resolve merge conflict between main and feature/advanced-feature'
[main a742afb] Resolve merge conflict between main and feature/advanced-feature
```

Una vez hallamos completado la corrección del conflicto generado por la completación entre ambas ramas procederemos a remover la rama de feature/advanced-feature, ya que sus cambios estan en main

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

```
bxco@bxco-NBD-WXX9:~/pregunta1$ git branch -d feature/advanced-feature  
Eliminada la rama feature/advanced-feature (era 2e56f53).  
bxco@bxco-NBD-WXX9:~/pregunta1$ git branch  
* main
```

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:

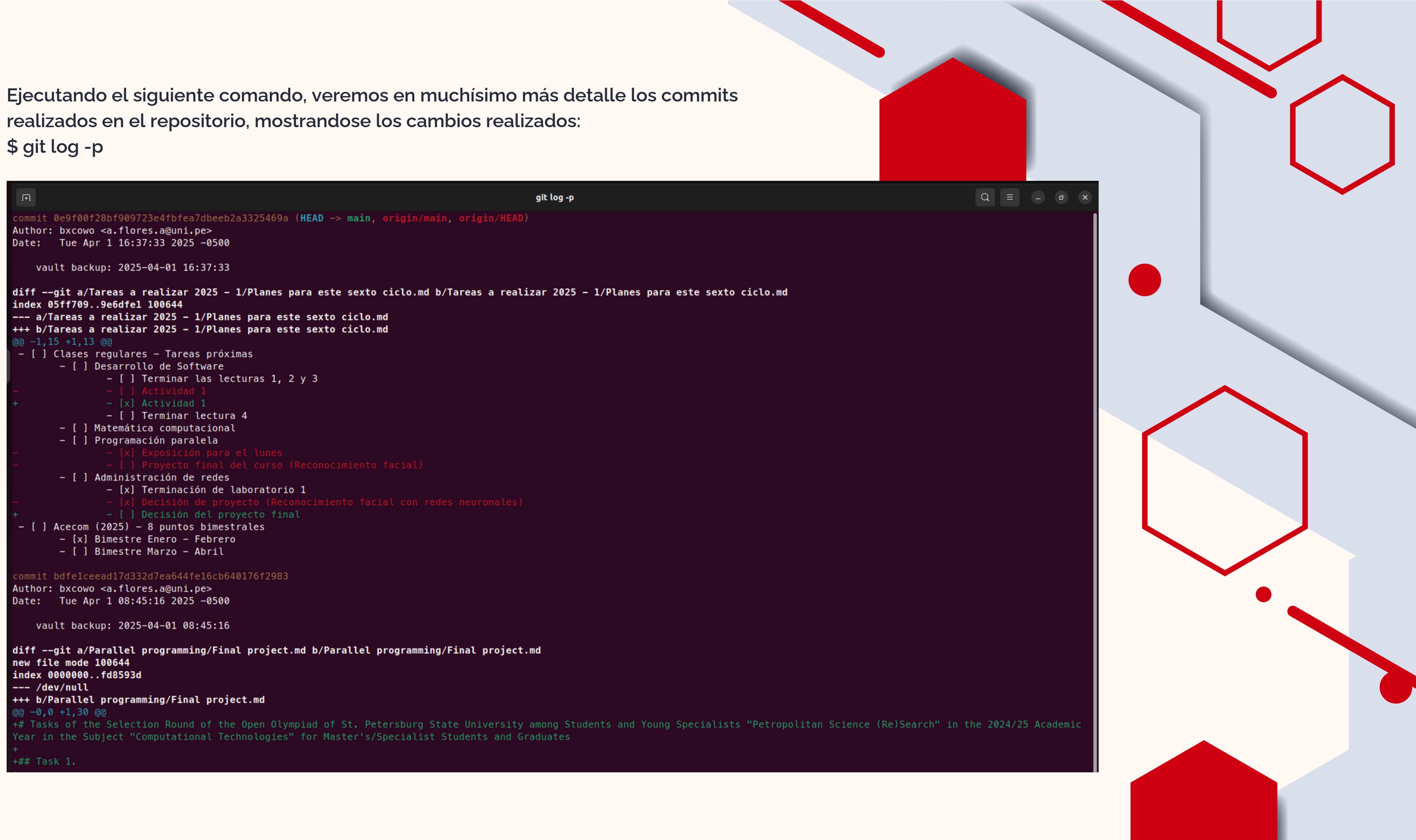
- > Computational Mathematics
- > Google Cloud Computing Foundations Acad...
- > knowledge
- Parallel programming
 - Final project
- Software Development
 - Actividad 1
 - Actividad 2
 - Lectura1
 - Lectura2
 - Lectura3
 - Lectura4
 - Lectura5
 - Lectura6
 - Lectura7
 - Lectura8
 - Lectura9
- Tareas 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

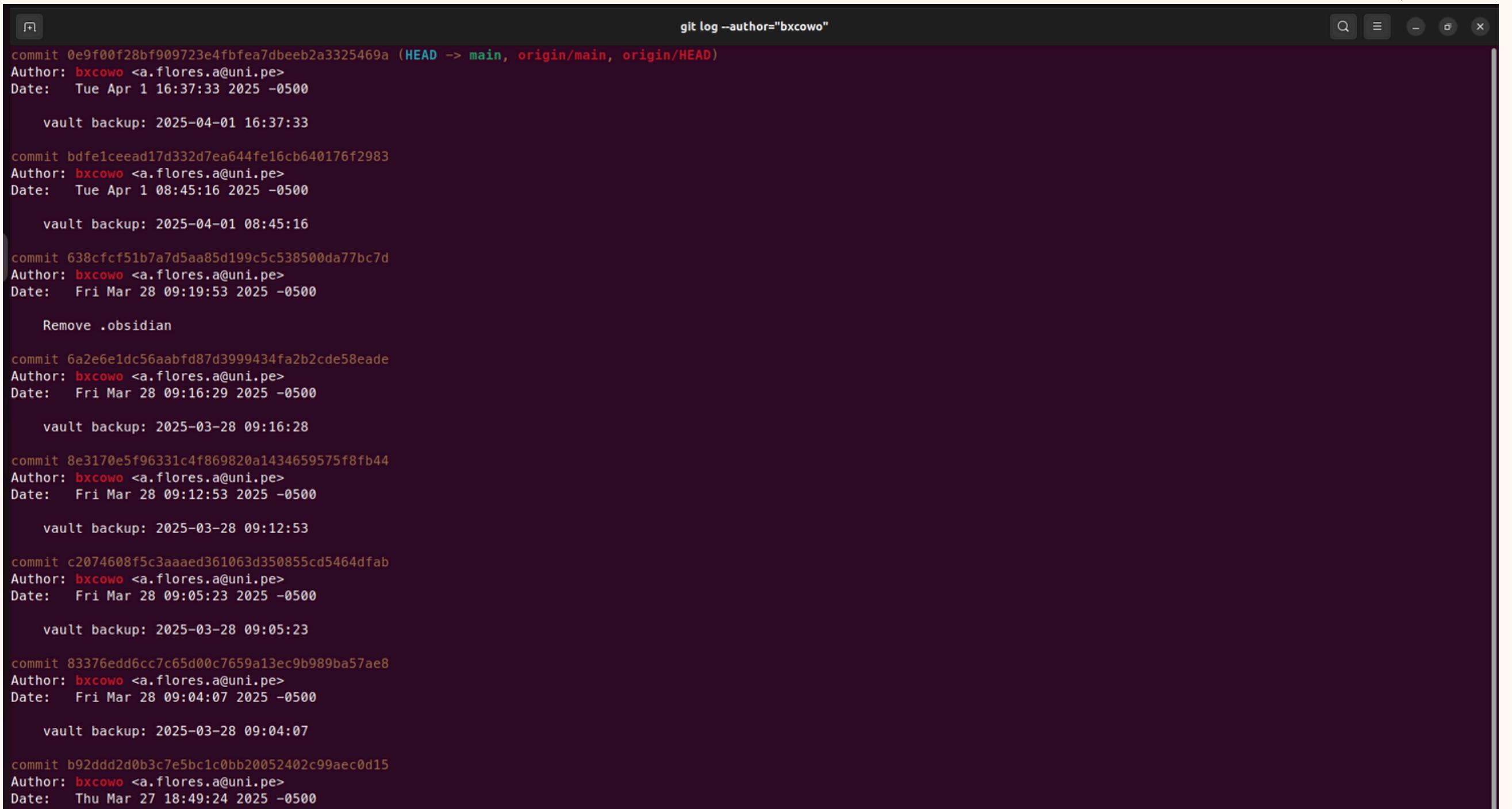
commit bdfe1ceead17d332d7ea644fe16cb640176f2983
Author: bxcowo <a.flores.a@uni.pe>
Date: Tue Apr 1 08:45:16 2025 -0500

    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(-)
```

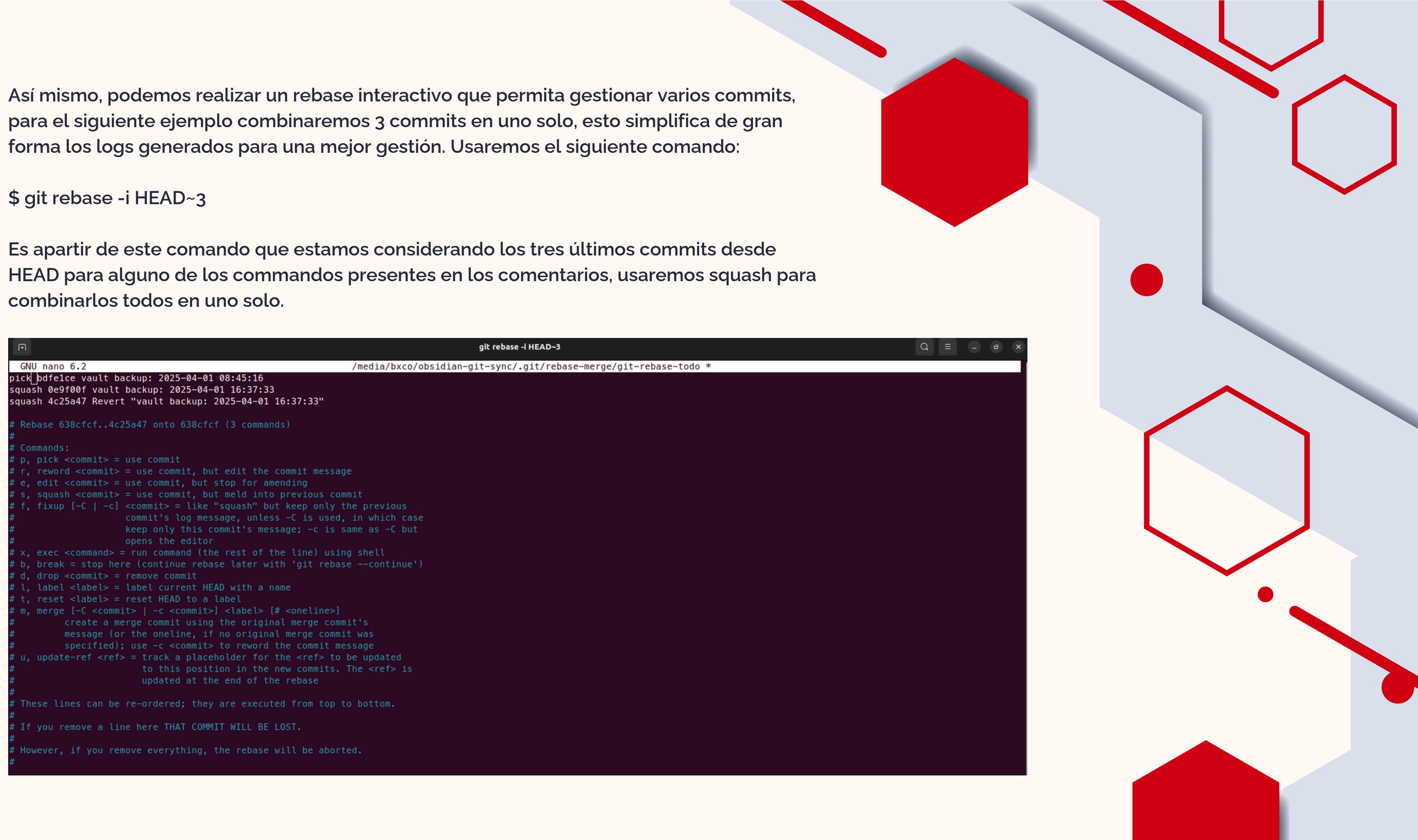
Ahora si verificasemos nuestro historial de commits usando \$ git log, notaremos los cambios hechos.

```
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
```

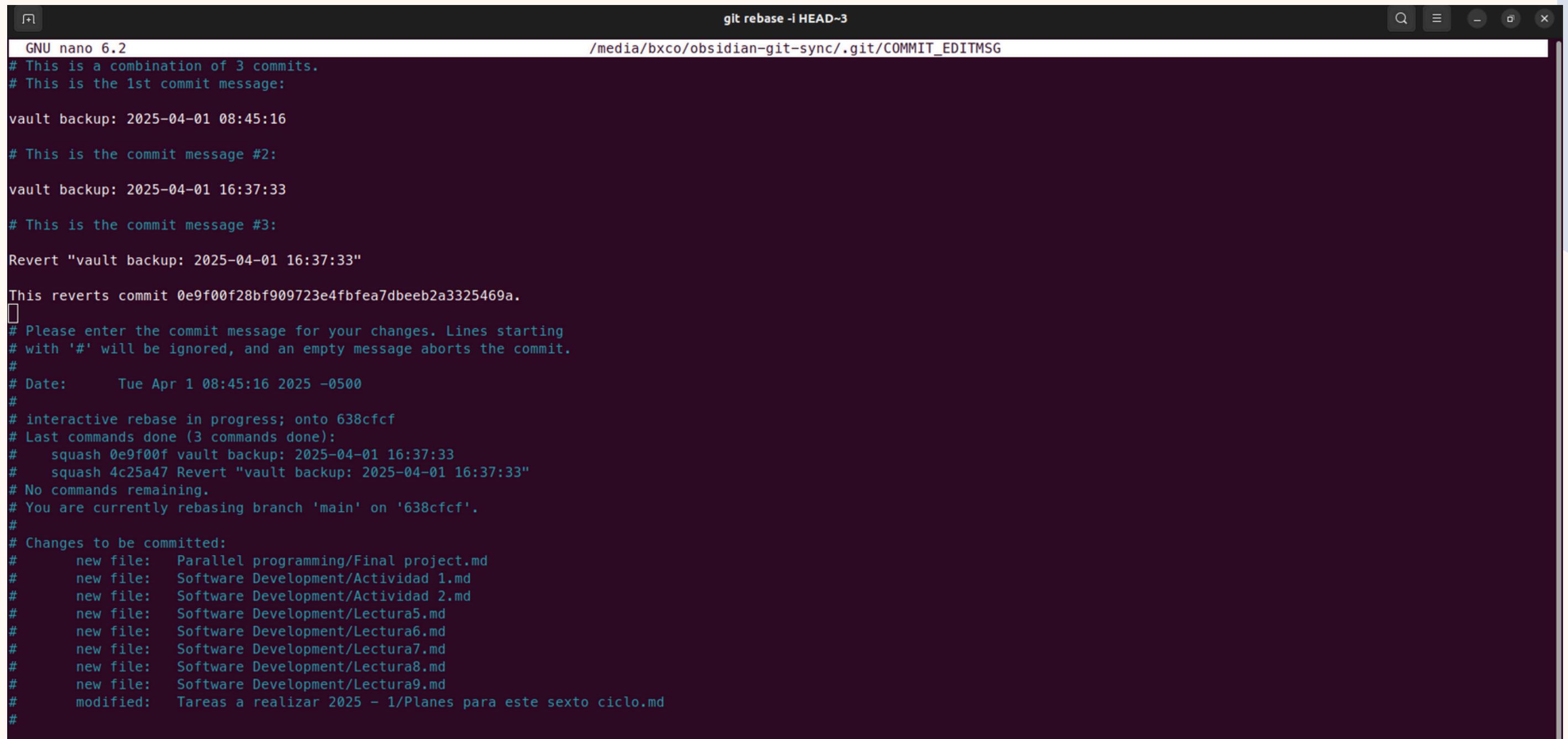
Es apartir de este comando que estamos considerando los tres últimos commits desde HEAD para alguno de los commandos presentes en los comentarios, usaremos squash para combinarlos todos en uno solo.



```
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

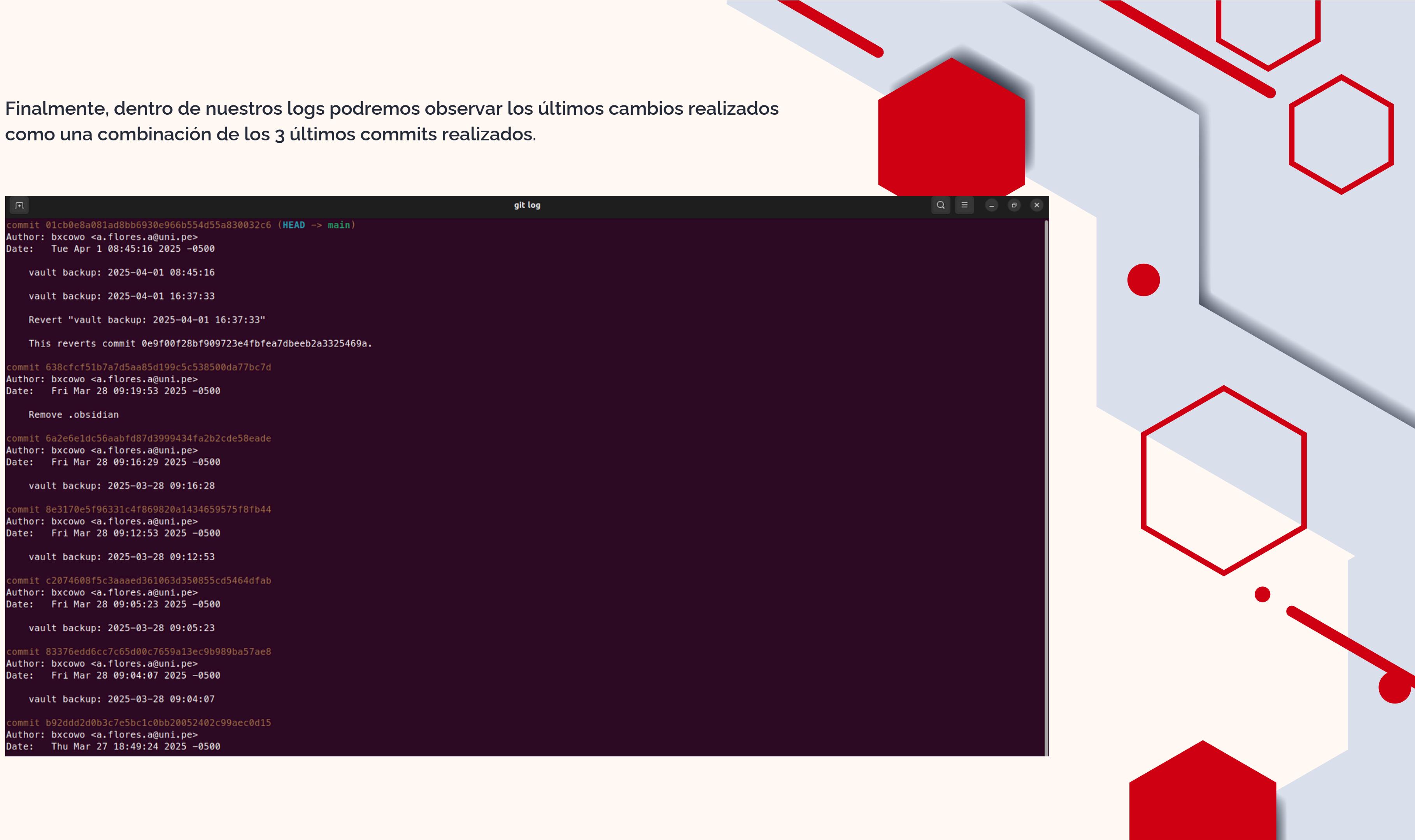
# This is the commit message #3:

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.

Dentro de nuestro repositorio de git tendremos un historial de todos los commits realizados, podemos visualizar esta información mediante el siguiente comando:

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

Ahora crearemos un nueva rama bugfix/rollback-feature desde el commit en el que se añadió el archivo main.py usando los siguientes comandos:

```
$ git branch bugfix/rollback-feature 540a26e  
$ git checkout bugfix/rollback-feature
```

Dentro de dicha rama realizaremos algunos cambios en main.py que nos simulen una corrección de errores y guardaremos todos los cambios realizados en dicha rama:

```
$ vim main.py  
$ git add main.py  
$ git commit -m 'Fix bug in rollback feature'
```

Cambiamos a la rama main donde intentaremos combinarla con aquella de corrección del error

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

Confirmaremos el commit de combinación que se nos aparece al ejecutar el comando de combinación.

```
bxco@bxco-NBD-WXX9:~/pregunta2$ git log --oneline  
b6adbf7 (HEAD -> main) Añadir año en README.md  
540a26e Añadimos archivo main.py  
19dec4f Añadir nueva linea en README.md  
7bda52d Commit inicial de la pregunta 3
```

```
bxco@bxco-NBD-WXX9:~/pregunta2$ git branch bugfix/rollback-feature 540a26e  
bxco@bxco-NBD-WXX9:~/pregunta2$ git checkout bugfix/rollback-feature  
Cambiado a rama 'bugfix/rollback-feature'  
bxco@bxco-NBD-WXX9:~/pregunta2$ git branch  
* bugfix/rollback-feature  
  main
```

```
def fix_bug():  
    print("Fixing the bug ...")  
    print("Done")  
~  
~
```

```
bxco@bxco-NBD-WXX9:~/pregunta2$ git add main.py  
bxco@bxco-NBD-WXX9:~/pregunta2$ git commit -m 'Fix bug in rollback feature'  
[bugfix/rollback-feature 627e6f4] Fix bug in rollback feature  
 1 file changed, 4 insertions(+)
```

```
GNU nano 6.2                               /home/bxco/pregunta2/.git/MERGE_MSG  
Merge branch 'bugfix/rollback-feature'  
# Por favor ingresa un mensaje de commit que explique por qué es necesaria esta fusión,  
# especialmente si esto fusiona un upstream actualizado en una rama de tópico.  
#  
# Lines starting with '#' will be ignored, and an empty message aborts  
# the commit.  
~
```

```
bxco@bxco-NBD-WXX9:~/pregunta2$ git checkout main  
Cambiado a rama 'main'  
bxco@bxco-NBD-WXX9:~/pregunta2$ git merge bugfix/rollback-feature  
Merge made by the 'ort' strategy.  
 main.py | 4 +++-  
 1 file changed, 3 insertions(+), 1 deletion(-)  
bxco@bxco-NBD-WXX9:~/pregunta2$
```

Una vez fusionadas ambas ramas de trabajo, podremos visualizar su historial de commits mediante un gráfico de términos ASCII mediante el siguiente comando:
\$ git log --graph --oneline --all

Entonces ahora que los cambios se registraron correctamente en la rama principal podemos eliminar la rama de bugfix/rollback-feature
\$ git branch -d bugfix/rollback-feature

```
bxco@bxco-NBD-WXX9:~/pregunta2$ git log --graph --oneline --all
*   9906ece (HEAD -> main) Merge branch 'bugfix/rollback-feature'
|\ 
| * 10249b1 (bugfix/rollback-feature) Again fix the bug in rollback feature
| * 627e6f4 Fix bug in rollback feature
* | b6adbf7 Añadir año en README.md
|/
* 540a26e Añadimos archivo main.py
* 19dec4f Añadir nueva linea en README.md
* 7bda52d Commit inicial de la pregunta 3
bxco@bxco-NBD-WXX9:~/pregunta2$ git branch -d bugfix/rollback-feature
Eliminada la rama bugfix/rollback-feature (era 10249b1).
```

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
```

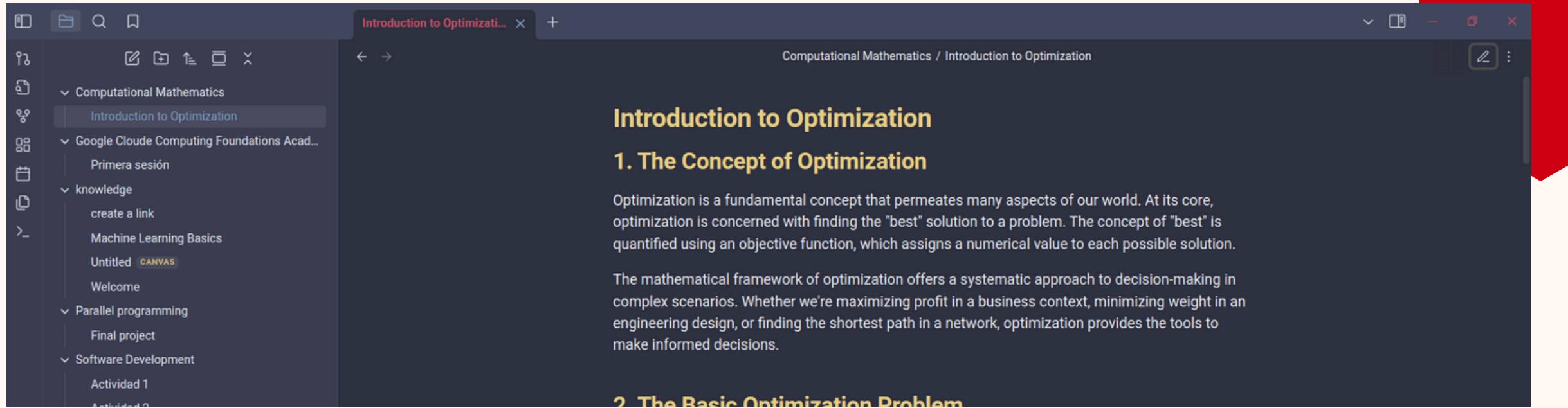
```
git reset --hard HEAD~1
HEAD is now at f03df8a vault backup: 2025-04-04 12:22:46
```

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

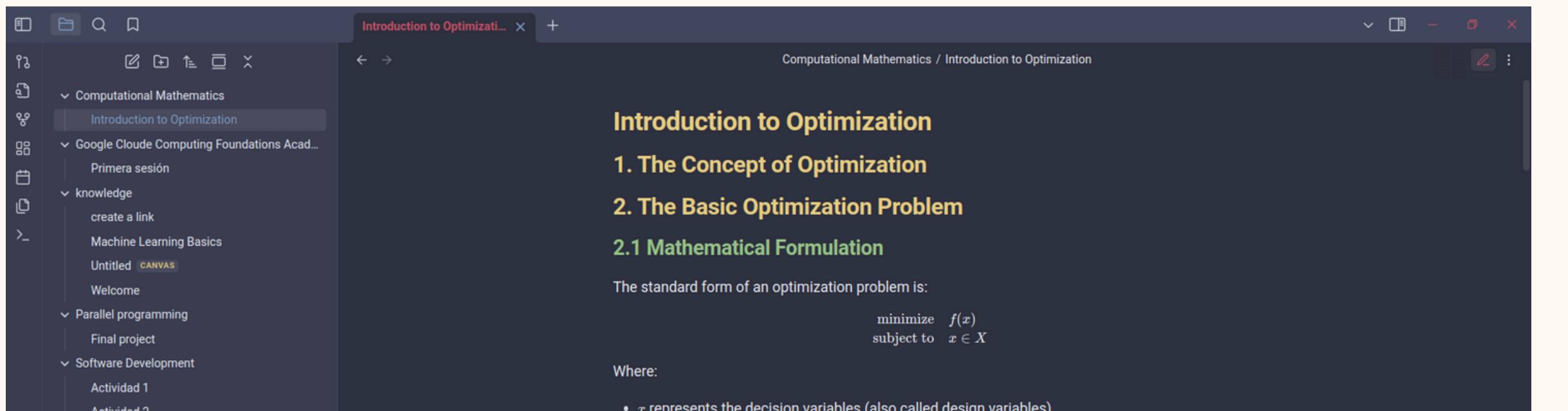
    vault backup: 2025-04-04 12:22:46
```

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



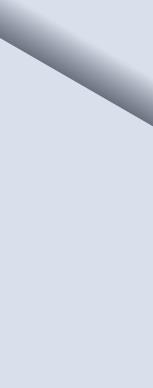
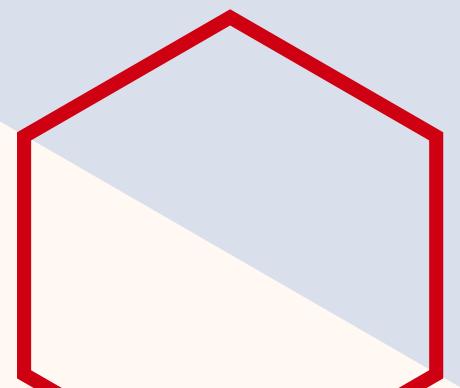
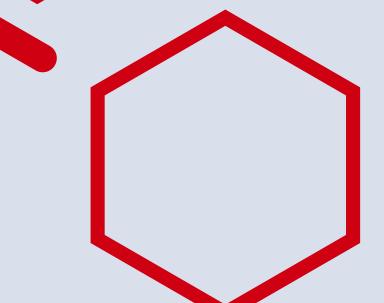
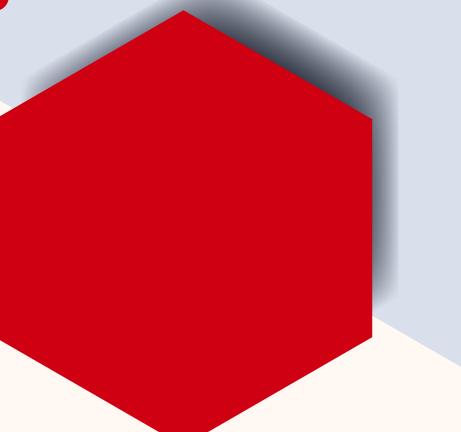
Ahora bien, se pueden revertir cambios registrados en nuestro repositorio local, pero estos procedimientos también pueden ser aplicados para aquellos que sean no registrados.

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
```



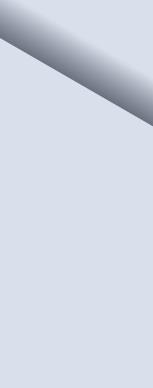
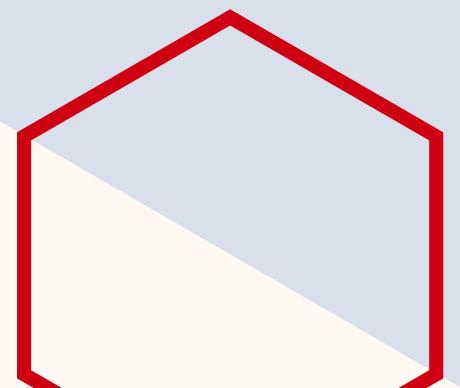
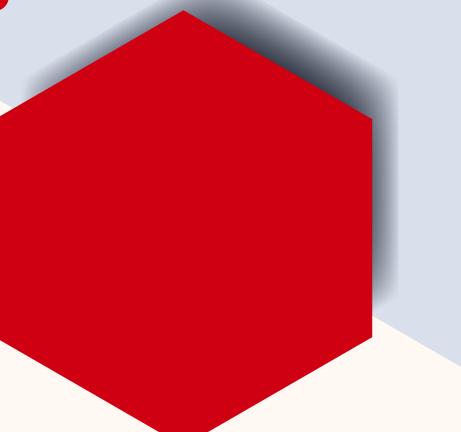
```
q ➔ /media/bxco/obsidian-git-sync ➔ 🐫 main !1
> git status
On branch main
Your branch is up to date with 'origin/main'.

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")
```

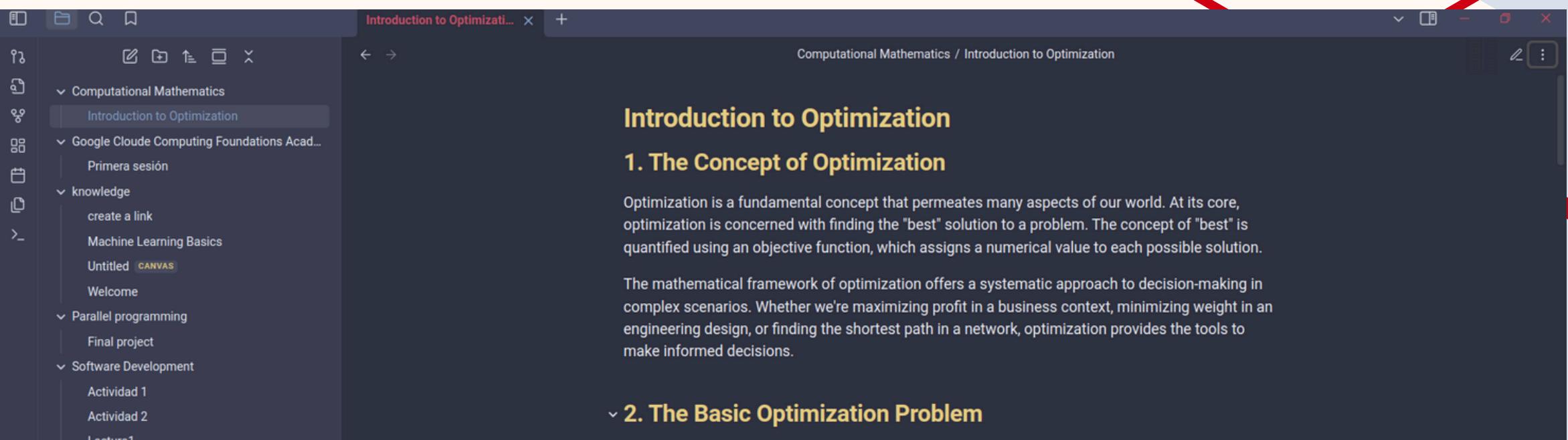
Para revertir dichos cambios sin necesariamente tener que utilizar 'Control + z', podemos utilizar el siguiente comando que especifique restaurar los cambios hechos en dicho archivo:

```
$ 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.



The screenshot shows the Obsidian application interface. On the left is a sidebar with a tree view of notes. The main area displays the content of the 'Introduction to Optimization' note. The title is 'Introduction to Optimization'. Below it is a section titled '1. The Concept of Optimization' with the following text: '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.' There is also a section titled '2. The Basic Optimization Problem'.

EJERCICIO 5



OBJETIVO

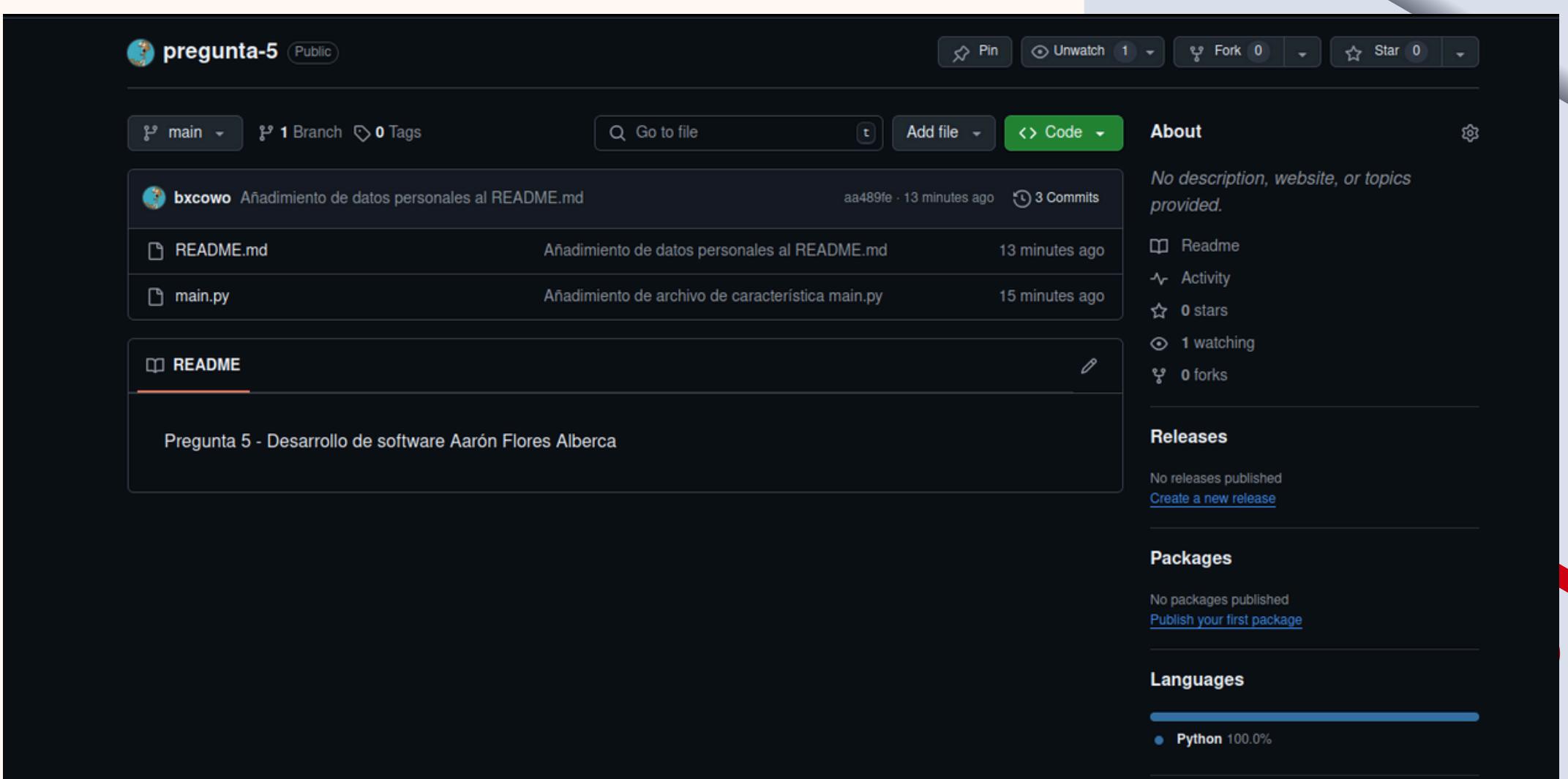
Simular un flujo de trabajo colaborativo utilizando ramas y pull requests.

Creamos un repositorio remoto dentro de GitHub con el cual tomar de receptor para la sincronización de un repositorio local. Una vez creado, ejecutamos desde la terminal de dicho repositorio local los siguientes comandos:

```
$ git remote add origin https://github.com/  
bxcowo/pregunta-5.git  
$ git push -u origin main
```

Ingresamos nuestras credenciales y se debería de visualizar los cambios en la página de github

```
bxco@bxco-NBD-WXX9:~/pregunta5$ git remote add origin https://github.com/bxcowo/pregunta-5.git  
bxco@bxco-NBD-WXX9:~/pregunta5$ git push -u origin main  
Username for 'https://github.com': bxcowo  
Password for 'https://bxcowo@github.com':  
Enumerando objetos: 9, listo.  
Contando objetos: 100% (9/9), listo.  
Compresión delta usando hasta 8 hilos  
Comprimiendo objetos: 100% (6/6), listo.  
Escribiendo objetos: 100% (9/9), 894 bytes | 894.00 KiB/s, listo.  
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)  
To https://github.com/bxcowo/pregunta-5.git  
 * [new branch]      main -> main  
rama 'main' configurada para rastrear 'origin/main'.
```



En nuestro repositorio cambiamos a una rama llamada 'feature/team-feature' y en el creamos un nuevo archivo llamado 'collaboration.py', finalmente lo commiteamos. Para esto ejecutemos los siguientes comandos:

```
$ git branch feature/team-feature  
$ git checkout feature/team-feature  
$ echo 'print("Collaboration is key!")' >  
collaboration.py  
$ git add collaboration.py  
$ git commit -m 'Añadimos archivo collaboration.py'
```

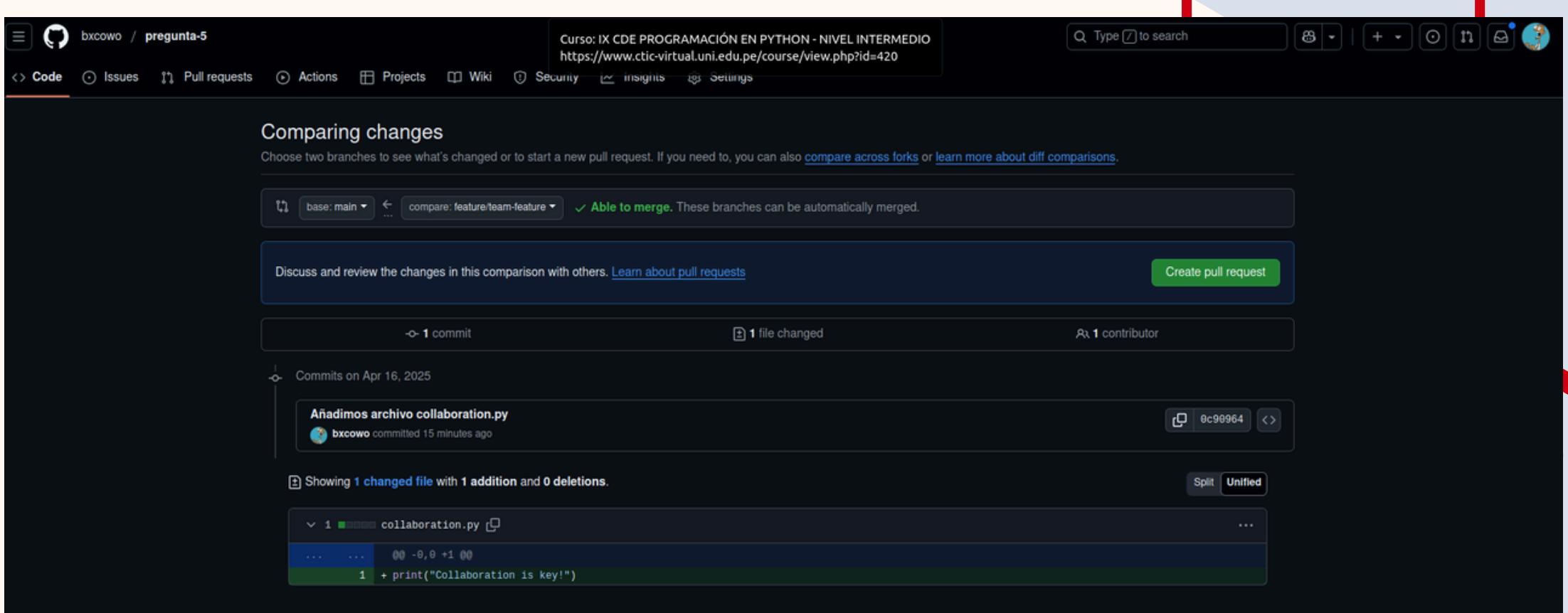
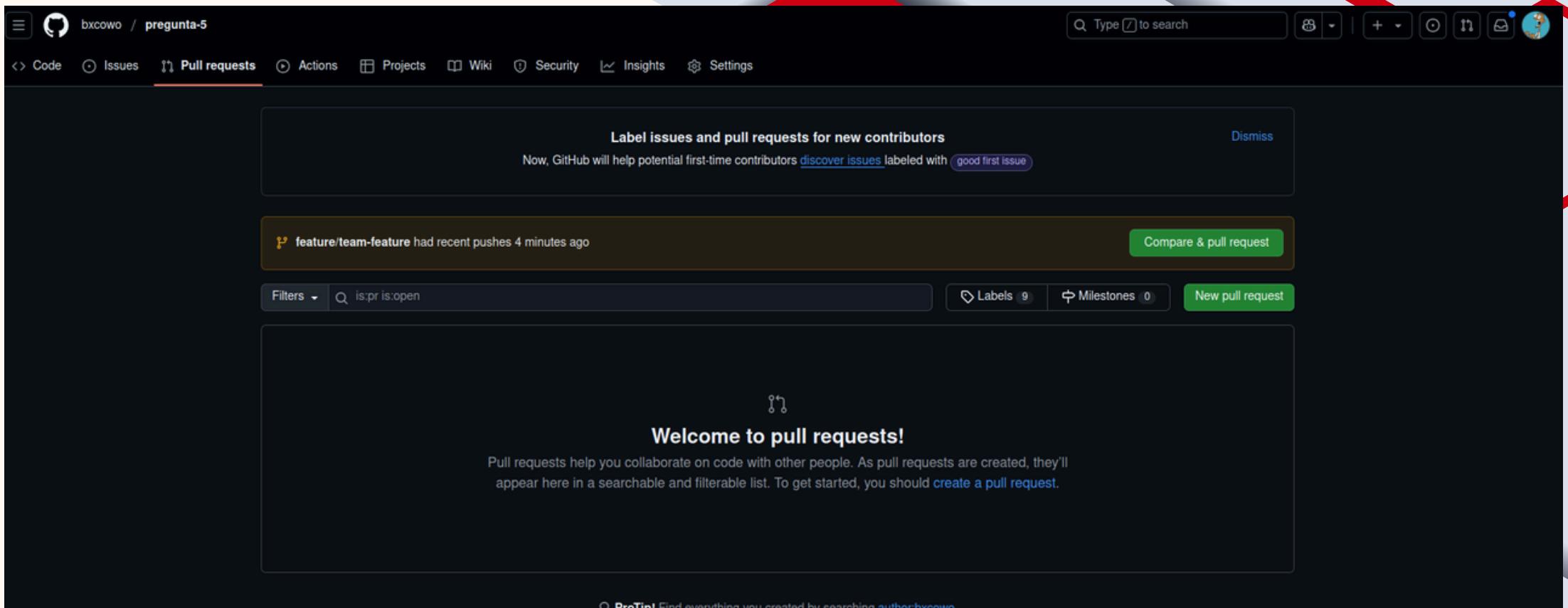
y los pusheamos dentro de dicha rama remota

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

```
bxco@bxco-NBD-WXX9:~/pregunta5$ git branch feature/team-feature  
bxco@bxco-NBD-WXX9:~/pregunta5$ git checkout feature/team-feature  
Cambiado a rama 'feature/team-feature'  
bxco@bxco-NBD-WXX9:~/pregunta5$ echo 'print("Collaboration is key!")' > collaboration.py  
bxco@bxco-NBD-WXX9:~/pregunta5$ git add collaboration.py  
bxco@bxco-NBD-WXX9:~/pregunta5$ git commit -m 'Añadimos archivo collaboration.py'  
[feature/team-feature 0c90964] Añadimos archivo collaboration.py  
 1 file changed, 1 insertion(+)  
 create mode 100644 collaboration.py  
bxco@bxco-NBD-WXX9:~/pregunta5$ git push origin feature/team-feature  
Username for 'https://github.com': bxcowo  
Password for 'https://bxcowo@github.com':  
Enumerando objetos: 4, listo.  
Contando objetos: 100% (4/4), listo.  
Compresión delta usando hasta 8 hilos  
Comprimiendo objetos: 100% (2/2), listo.  
Escribiendo objetos: 100% (3/3), 359 bytes | 359.00 KiB/s, listo.  
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)  
remote:  
remote: Create a pull request for 'feature/team-feature' on GitHub by visiting:  
remote:     https://github.com/bxcowo/pregunta-5/pull/new/feature/team-feature  
remote:  
To https://github.com/bxcowo/pregunta-5.git  
 * [new branch]      feature/team-feature -> feature/team-feature
```

Ahora podemos ingresar a la plataforma de GitHub para inicializar un Pull Request (PR) desde la sección de Pull Request en el repositorio. En él seleccionamos que deseamos realizar un merge desde la rama main hacia la feature/team-future

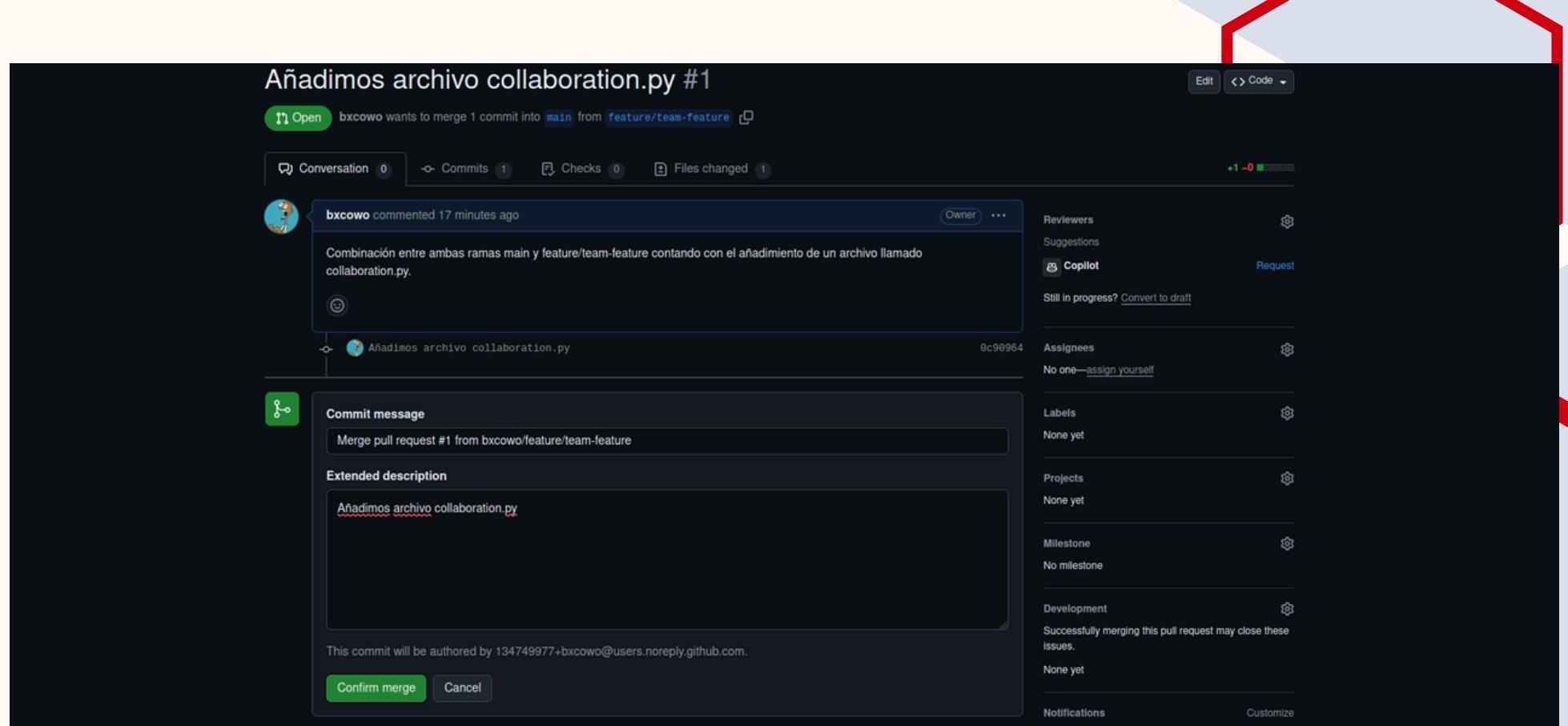
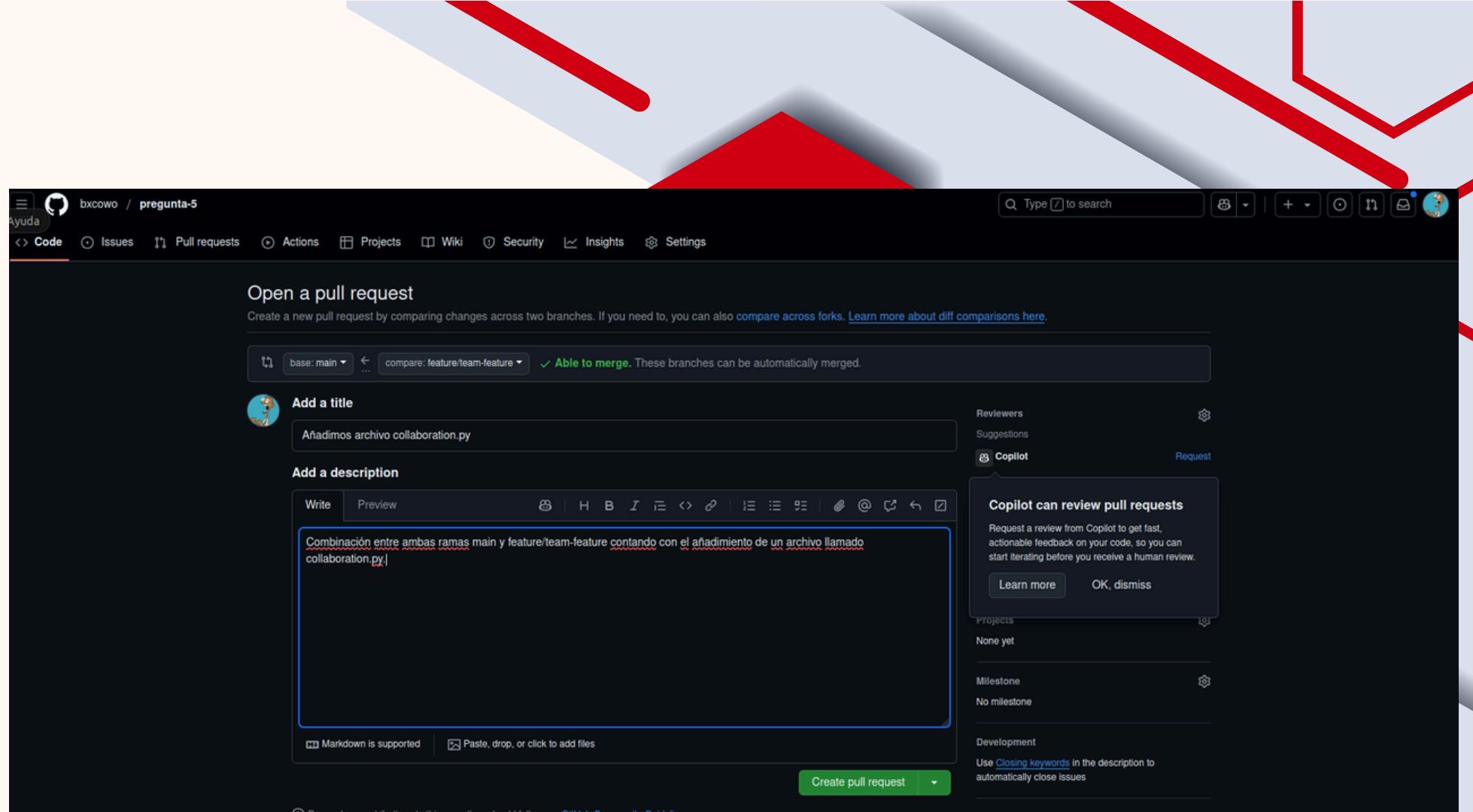
Cuando ingresemos, se nos aparecerá una ventana indicando los cambios entre ambas ramas con las que deseemos realizar el merge y si resulta sin conflictos



Seguimos añadiendo una descripción al pull request, es decir, indicando los cambios y razones de merge.

Una vez concluido entonces podremos visualizar que se nos da la opción de realizar un merge request donde también necesitaremos dar una descripción de este.

Finalmente le damos a 'confirm merge' para confirmar la acción



Si ingresamos a continuación, dentro de nuestra rama main se encuentran los archivos del merge realizado junto con los que se tenían en un inicio.

Para ver los cambios en nuestro repositorio local usamos:

```
$ git pull
```

Ahora que estan mergeados no necesitamos la rama feature/team-feature, por lo que eliminaremos dicha rama mediante los siguientes comandos:

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

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

```
bxco@bxco-NBD-WXX9:~/pregunta5$ git pull
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Desempaquetando objetos: 100% (1/1), 932 bytes | 932.00 KiB/s, listo.
Desde https://github.com/bxcowo/pregunta-5
      aa489fe..3725fef  main          -> origin/main
Actualizando aa489fe..3725fef
Fast-forward
  collaboration.py | 1 +
  1 file changed, 1 insertion(+)
  create mode 100644 collaboration.py
bxco@bxco-NBD-WXX9:~/pregunta5$ git branch -d feature/team-feature
Eliminada la rama feature/team-feature (era 0c90964).
bxco@bxco-NBD-WXX9:~/pregunta5$ git push origin --delete feature/team-feature
Username for 'https://github.com': bxcowo
Password for 'https://bxcowo@github.com':
To https://github.com/bxcowo/pregunta-5.git
 - [deleted]           feature/team-feature
```

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.

Realizamos cambios dentro de main.py como una nueva feature ejecutando los siguientes comandos:

```
$ echo 'print("Cherry pick this")' >> main.py  
$ git add main.py  
$ git commit -m 'Añadimos ejemplo de cherry pick'
```

Verificamos nuestro log de commits usando:

```
$ git log
```

Luego generamos una nueva rama a la cual aplicar dicho commit en particular mediante los siguientes comandos:

```
$ git branch feature/cherry-pick  
1643f1e5c0d88edfc080ea48e3a7dfa24cf9e4df  
$ git checkout feature/cherry-pick  
$ git log
```

```
bxco@bxco-NBD-WXX9:~/pregunta6$ echo 'print("Cherry pick this")' >> main.py  
bxco@bxco-NBD-WXX9:~/pregunta6$ git add main.py  
bxco@bxco-NBD-WXX9:~/pregunta6$ git commit -m 'Añadimos ejemplo de cherry pick'  
[main 3db74c5] Añadimos ejemplo de cherry pick  
 1 file changed, 1 insertion(+)  
bxco@bxco-NBD-WXX9:~/pregunta6$ git branch feature/cherry-pick  
bxco@bxco-NBD-WXX9:~/pregunta6$ git checkout feature/cherry-pick  
Cambiado a rama 'feature/cherry-pick'
```

```
bxco@bxco-NBD-WXX9:~/pregunta6$ git log  
commit 3db74c555f9afbdb15ac6d02f2e629153d65d74e (HEAD -> main)  
Author: bxcowo <a.flores.a@uni.pe>  
Date:   Wed Apr 16 09:22:44 2025 -0500  
  
    Añadimos ejemplo de cherry pick  
  
commit 1643f1e5c0d88edfc080ea48e3a7dfa24cf9e4df  
Author: bxcowo <a.flores.a@uni.pe>  
Date:   Wed Apr 16 09:19:12 2025 -0500  
  
    Edición de README.md al añadir nombre  
  
commit e3db5bd9c4f08242ba4d97edbc0187beea116d12  
Author: bxcowo <a.flores.a@uni.pe>  
Date:   Wed Apr 16 09:18:36 2025 -0500  
  
    Añadimiento de archivo main.py  
  
commit 356420e482e3c3e2322c34b9c3ca47680c4b25b5  
Author: bxcowo <a.flores.a@uni.pe>  
Date:   Wed Apr 16 09:16:41 2025 -0500  
  
    Commit inicial del repositorio de la pregunta 6  
bxco@bxco-NBD-WXX9:~/pregunta6$ git branch feature/cherry-pick 1643f1e5c0d88edfc080ea48e3a7dfa24cf9e4df  
bxco@bxco-NBD-WXX9:~/pregunta6$ git checkout feature/cherry-pick  
Cambiado a rama 'feature/cherry-pick'  
bxco@bxco-NBD-WXX9:~/pregunta6$ git log  
commit 1643f1e5c0d88edfc080ea48e3a7dfa24cf9e4df (HEAD -> feature/cherry-pick)  
Author: bxcowo <a.flores.a@uni.pe>  
Date:   Wed Apr 16 09:19:12 2025 -0500  
  
    Edición de README.md al añadir nombre  
  
commit e3db5bd9c4f08242ba4d97edbc0187beea116d12  
Author: bxcowo <a.flores.a@uni.pe>  
Date:   Wed Apr 16 09:18:36 2025 -0500  
  
    Añadimiento de archivo main.py  
  
commit 356420e482e3c3e2322c34b9c3ca47680c4b25b5  
Author: bxcowo <a.flores.a@uni.pe>  
Date:   Wed Apr 16 09:16:41 2025 -0500  
  
    Commit inicial del repositorio de la pregunta 6
```

Ahora probaremos a realizar el comando cherry-pick desde esta nueva rama como

```
$ git cherry-pick 3db74c555f9afbdb15ac6d02f2e629153d65d74e
```

```
$ git log
```

Así habremos aplicado un commit en específico para una rama diferente desde la que se aplicó.

```
bxco@bxco-NBD-WXX9:~/pregunta6$ git cherry-pick 3db74c555f9afbdb15ac6d02f2e629153d65d74e
[feature/cherry-pick 4546839] Añadimos ejemplo de cherry pick
Date: Wed Apr 16 09:22:44 2025 -0500
1 file changed, 1 insertion(+)
bxco@bxco-NBD-WXX9:~/pregunta6$ git log
commit 4546839ac502809d92a60457d04406cfbd2c6743 (HEAD -> feature/cherry-pick)
Author: bxcowo <a.flores.a@uni.pe>
Date:   Wed Apr 16 09:22:44 2025 -0500

    Añadimos ejemplo de cherry pick

commit 1643f1e5c0d88edfc080ea48e3a7dfa24cf9e4df
Author: bxcowo <a.flores.a@uni.pe>
Date:   Wed Apr 16 09:19:12 2025 -0500

    Edición de README.md al añadir nombre

commit e3db5bd9c4f08242ba4d97edbc0187beea116d12
Author: bxcowo <a.flores.a@uni.pe>
Date:   Wed Apr 16 09:18:36 2025 -0500

    Añadimientode archivo main.py

commit 356420e482e3c3e2322c34b9c3ca47680c4b25b5
Author: bxcowo <a.flores.a@uni.pe>
Date:   Wed Apr 16 09:16:41 2025 -0500

Commit inicial del repositorio de la pregunta 6
```

Intentemos ahora realizar diferentes cambios en dicha rama sin necesariamente guardarlos en nuestro repositorio

```
$ echo 'Este cambio es stashed' >> main.py  
$ git status
```

Ahora podremos guardar estos pequeños cambios mediante el siguiente comando:

```
$ git stash
```

Ahora podríamos incluso realizar diferentes cambios en otros archivos, como por ejemplo:

```
$ echo 'Cambio nuevo' >> README.md  
$ echo 'print("Nuevos cambios")' >> main.py
```

```
bxco@bxco-NBD-WXX9:~/pregunta6$ git branch  
* feature/cherry-pick  
  main  
bxco@bxco-NBD-WXX9:~/pregunta6$ echo 'Este cambio es stashed' >> main.py  
bxco@bxco-NBD-WXX9:~/pregunta6$ git status  
En la rama feature/cherry-pick  
Cambios no rastreados para el commit:  
  (usa "git add <archivo>..." para actualizar lo que será confirmado)  
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)  
    modificados:    main.py  
  
Archivos sin seguimiento:  
  (usa "git add <archivo>..." para incluirlo a lo que será confirmado)  
    .main.py.swp  
  
sin cambios agregados al commit (usa "git add" y/o "git commit -a")  
bxco@bxco-NBD-WXX9:~/pregunta6$ git stash  
Directorio de trabajo y estado de índice WIP on feature/cherry-pick: 4546839 Añadimos ejemplo de cherry pick guardados  
bxco@bxco-NBD-WXX9:~/pregunta6$ git status  
En la rama feature/cherry-pick  
Archivos sin seguimiento:  
  (usa "git add <archivo>..." para incluirlo a lo que será confirmado)  
    .main.py.swp  
  
no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerles seguimiento)
```

```
bxco@bxco-NBD-WXX9:~/pregunta6$ echo 'Cambio nuevo' >> README.md  
bxco@bxco-NBD-WXX9:~/pregunta6$ echo 'print("Nuevos cambios")' >> main.py  
bxco@bxco-NBD-WXX9:~/pregunta6$ cat README.md  
Pregunta 6 - Desarrollo de software  
Aarón Flores Alberca  
Cambio nuevo  
bxco@bxco-NBD-WXX9:~/pregunta6$ cat main.py  
print("Hello world!")  
print("Cherry pick this")  
print("Nuevos cambios")
```

Ahora guardamos nuestros cambios realizados despues del stash ejecutando los comandos:

```
$ git add .  
$ git commit -m 'Commit de cambios en stashed'
```

Se nos informa que hubo un error en estos debido a que durante ambas etapas se modificó el archivo main.py, así que tendremos que solucionar el error nosotros mismos:

```
$ vim main.py
```

Nuevamente commitemos para guardar nuestros cambios finales mediante:

```
$ git add .  
$ git commit -m 'Commit de corrección en main.py  
después del stash'
```

```
bxco@bxco-NBD-WXX9:~/pregunta6$ git add .  
bxco@bxco-NBD-WXX9:~/pregunta6$ git commit -m 'Commit de cambios realizados despues del stash'  
[feature/cherry-pick 5b1126d] Commit de cambios realizados despues del stash  
 3 files changed, 2 insertions(+)  
  create mode 100644 .main.py.swp  
bxco@bxco-NBD-WXX9:~/pregunta6$ git stash pop  
Auto-fusionando main.py  
CONFLICTO (contenido): Conflicto de fusión en main.py  
En la rama feature/cherry-pick  
Rutas no fusionadas:  
  (usa "git restore --staged <archivo>..." para sacar del área de stage)  
  (usa "git add <archivo>..." para marcar una resolución)  
    modificados por ambos: main.py  
  
sin cambios agregados al commit (usa "git add" y/o "git commit -a")  
La entrada de stash se guardó en caso de ser necesario nuevamente.
```

```
bxco@bxco-NBD-WXX9:~/pregunta6$ vim main.py  
bxco@bxco-NBD-WXX9:~/pregunta6$ git add .  
bxco@bxco-NBD-WXX9:~/pregunta6$ git commit -m 'Commit de corrección en main.py despues del stash'  
[feature/cherry-pick 0ac4f88] Commit de corrección en main.py despues del stash  
 2 files changed, 1 insertion(+), 1 deletion(-)  
  delete mode 100644 .main.py.swp  
bxco@bxco-NBD-WXX9:~/pregunta6$ git stash pop  
En la rama feature/cherry-pick  
nada para hacer commit, el árbol de trabajo está limpio  
Descartado refs/stash@{0} (00b83090303c647db46b7932172f36e4864b4a43)  
bxco@bxco-NBD-WXX9:~/pregunta6$ cat README.md  
Pregunta 6 - Desarrollo de software  
Aarón Flores Alberca  
Cambio nuevo  
bxco@bxco-NBD-WXX9:~/pregunta6$ cat main.py  
print("Hello world!")  
print("Cherry pick this")  
Este cambio es stashed
```

Finalmente tendremos los cambios actualizados, podemos verificando haciendo uso de:
\$git log

```
Este cambio es stashed
bxco@bxco-NBD-WXX9:~/pregunta6$ git log
commit 0ac4f884a7ec02468db728c1d36124a4ed9de054 (HEAD -> feature/cherry-pick)
Author: bxcowo <a.flores.a@uni.pe>
Date:   Mon Apr 21 01:52:58 2025 -0500

    Commit de corrección en main.py despues del stash

commit 5b1126d8e79a6c1b0e501a49bbc2852bdd50eb0a
Author: bxcowo <a.flores.a@uni.pe>
Date:   Mon Apr 21 01:50:25 2025 -0500

    Commit de cambios realizados despues del stash

commit 4546839ac502809d92a60457d04406cfbd2c6743
Author: bxcowo <a.flores.a@uni.pe>
Date:   Wed Apr 16 09:22:44 2025 -0500

    Añadimos ejemplo de cherry pick

commit 1643f1e5c0d88edfc080ea48e3a7dfa24cf9e4df
Author: bxcowo <a.flores.a@uni.pe>
Date:   Wed Apr 16 09:19:12 2025 -0500

    Edición de README.md al añadir nombre

commit e3db5bd9c4f08242ba4d97edbc0187beea116d12
Author: bxcowo <a.flores.a@uni.pe>
Date:   Wed Apr 16 09:18:36 2025 -0500

    Añadimiento de archivo main.py

commit 356420e482e3c3e2322c34b9c3ca47680c4b25b5
Author: bxcowo <a.flores.a@uni.pe>
Date:   Wed Apr 16 09:16:41 2025 -0500

    Commit inicial del repositorio de la pregunta 6
```



**MUCHAS
GRACIAS**