

Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0117 – Programación Bajo Plataformas Abiertas.
III-ciclo 2023

Laboratorio 5

GIT

Ximena Céspedes Quesada
B91969

Profesor:
Julián Gairaud

10/02/23

Índice

1. Fork	1
2. Commits	2
2.1. Commit Gato1	2
2.2. Creacion del Gato3	3
2.3. Rama “feature” al repositorio remoto	4
3. Cambios en la rama main	4
4. Inclusión de la rama feature en main	5
5. Conflictos	6
6. Resolución de conflictos	7
7. Cambios en el repositorio remoto	7
8. Manual git log	9
9. Archivo gatos.py	9
10.Repositorio local	10
11.Push	10

Índice de figuras

1.	Creando un “fork” a un repositorio personal.	1
2.	Clon del repositorio a la computadora.	1
3.	Creación de la rama feature	2
4.	Cambio de nombre gato 1 en feature	2
5.	Cambio a etapa staged	2
6.	Creación del primer commit	3
7.	Creación del gato3 en feature	3
8.	Creación del segundo commit	4
9.	Rama feature en el repositorio remoto	4
10.	Comprobación de la rama feature en el repositorio remoto	4
11.	Cambio de nombre gato1 desde main	5
12.	Creación del tercer commit	5
13.	Inclusión de feature en main	6
14.	Conflicto entre ramas	6
15.	Solución del conflicto	7
16.	Aplicación de la solución.	7
17.	Cambios al repositorio remoto.	8
18.	Verificación de que los cambios hayan sido incluidos al repositorio remoto.	8
19.	Comando “git log –oneline –graph”	8
20.	Comando “–graph”	9
21.	Comando “–oneline”	9
22.	Para que fue diseñado “–oneline”	9
23.	Archivo gatos.py	9
24.	Creación del pdf.	10
25.	Push de la rama main al repositorio remoto.	10
26.	Verificación de los cambios en el repositorio remoto.	11

1. Fork

Primeramente, para realizar un “fork” de un repositorio específico, es necesario estar en el dicho repositorio, para seguidamente, utilizando la opción “fork” crear una “copia” de dicho repositorio en un repositorio personal, como se observa en la figura 1

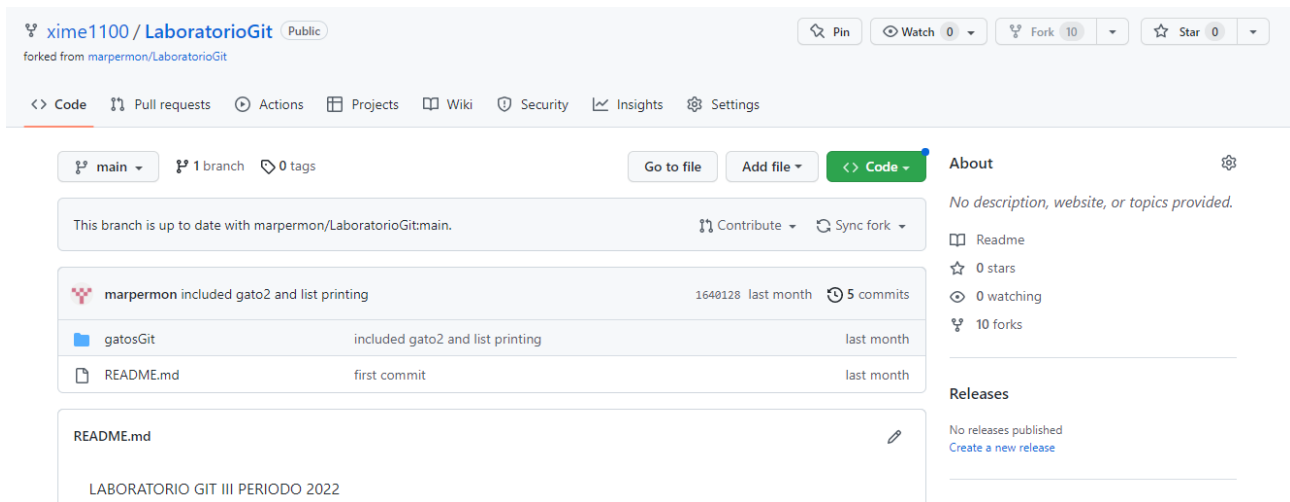


Figura 1: Creando un “fork” a un repositorio personal.

Seguidamente, mediante el comando “git clone”, se clona el repositorio a la computadora, tal y como se muestra en la figura 2.

```
ximenacq@ximena-vm:~$ git clone git@github.com:xime1100/LaboratorioGit.git
Cloning into 'LaboratorioGit'...
Warning: Permanently added the ECDSA host key for IP address '140.82.112.4' to the list of known hosts.
remote: Enumerating objects: 19, done.
remote: Counting objects: 100% (19/19), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 19 (delta 2), reused 19 (delta 2), pack-reused 0
Receiving objects: 100% (19/19), done.
Resolving deltas: 100% (2/2), done.
```

Figura 2: Clon del repositorio a la computadora.

2. Commits

Ahora bien, se crea una nueva rama mediante el comando “git branch < nombre >”, seguido del comando “git checkout”, para cambiar de la rama main a la nueva rama. Como se observa en la figura 3

```
ximenacq@ximena-vm:~/LaboratorioGit$ git branch feature
ximenacq@ximena-vm:~/LaboratorioGit$ git checkout feature
Switched to branch 'feature'
```

Figura 3: Creación de la rama feature

2.1. Commit Gato1

Primeramente, como se observa en la figura 4, mediante el editor de texto nano se modifica el archivo “gatosGit/gatos.py”. Seguidamente como se muestra en la figura 5, con el comando “git status”, se observa que el archivo ha sido modificado pero que este esta en “unstaged”, por lo tanto para llevarlo a la etapa de staged, se utiliza el comando “git add”

```
ximenacq@ximena-vm: ~/LaboratorioGit
GNU nano 4.8      gatosGit/gatos.py
import funcionesTarea

gato1 = funcionesTarea.gato()
gato1.nombre = "Campanita"
gato1.color = "negro"
gato1.edad = 4
gato1.aniadirlista()
```

Figura 4: Cambio de nombre gato 1 en feature

```
ximenacq@ximena-vm:~/LaboratorioGit$ nano gatosGit/gatos.py
ximenacq@ximena-vm:~/LaboratorioGit$ git status
On branch 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:   gatosGit/gatos.py

no changes added to commit (use "git add" and/or "git commit -a")
ximenacq@ximena-vm:~/LaboratorioGit$ git add gatosGit/gatos.py
ximenacq@ximena-vm:~/LaboratorioGit$ git status
On branch feature
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   gatosGit/gatos.py
```

Figura 5: Cambio a etapa staged

Luego, se procede a realizar el commit utilizando el comando “git commit -m < nombre >”, seguidamente, mediante el comando “git log”, se verifica que el commit se haya creado de manera correcta. Como se muestra en la figura 6

```
ximenacq@ximena-vm:~/LaboratorioGit$ git commit -m "Cambio de nombre del objeto gato1"
[feature b7649bd] Cambio de nombre del objeto gato1
1 file changed, 1 insertion(+), 1 deletion(-)
ximenacq@ximena-vm:~/LaboratorioGit$ git log
commit b7649bd200f64e8b598e329413cd26bbe2b19c73 (HEAD -> feature)
Author: xime1100 <ximena.cespedesquesada@ucr.ac.cr>
Date: Thu Feb 9 19:52:30 2023 -0600

    Cambio de nombre del objeto gato1
```

Figura 6: Creación del primer commit

2.2. Creacion del Gato3

Al igual que el punto anterior, se utiliza el editor nano para realizar la modificación en el archivo. Como se muestra en la figura 7.

```
ximenacq@ximena-vm: ~/LaboratorioGit
GNU nano 4.8 gatosGit/gatos.py
import funcionesTarea

gato1 = funcionesTarea.gato()
gato1.nombre = "Campanita"
gato1.color = "negro"
gato1.edad = 4
gato1.aniadirlista()

gato2 = funcionesTarea.gato()
gato2.nombre = "Pelusa"
gato2.color = "cafe"
gato2.edad = 3
gato2.aniadirlista()

gato3 = funcionesTarea.gato()
gato3.nombre = "Jimmy"
gato3.color = "gris"
gato3.edad = 1
gato3.aniadirlista()

funcionesTarea.printlista()
```

Figura 7: Creación del gato3 en feature

Seguidamente, se siguen los mismos pasos del punto anterior, como se observa en la figura 8

```
ximenacq@ximena-vm:~/LaboratorioGit$ nano gatosGit/gatos.py
ximenacq@ximena-vm:~/LaboratorioGit$ git add gatosGit/gatos.py
ximenacq@ximena-vm:~/LaboratorioGit$ git commit -m "Creacion de objeto gato3"
[feature d57fc66] Creacion de objeto gato3
1 file changed, 6 insertions(+)
ximenacq@ximena-vm:~/LaboratorioGit$ git log
commit d57fc66e55ab07f6de3648fe4077a76069ca0702 (HEAD -> feature)
Author: xime1100 <ximena.cespedesquesada@ucr.ac.cr>
Date: Thu Feb 9 19:55:08 2023 -0600

    Creacion de objeto gato3
```

Figura 8: Creación del segundo commit

2.3. Rama “feature” al repositorio remoto

Ahora bien, para poder subir la rama feature a el repositorio remoto, se utiliza el comando “git push”, como se muestra en la figura 9, donde seguido de este comando, indica que lleve los archivos desde la rama feature de la computadora hasta la rama feature en el repositorio remoto.

```
ximenacq@ximena-vm:~/LaboratorioGit$ git push origin feature:feature
```

Figura 9: Rama feature en el repositorio remoto

Seguidamente, se corrobora que la rama se haya subido de manera exitosa y que contenga los dos commits creados anteriormente, tal y como se observa en la figura 10

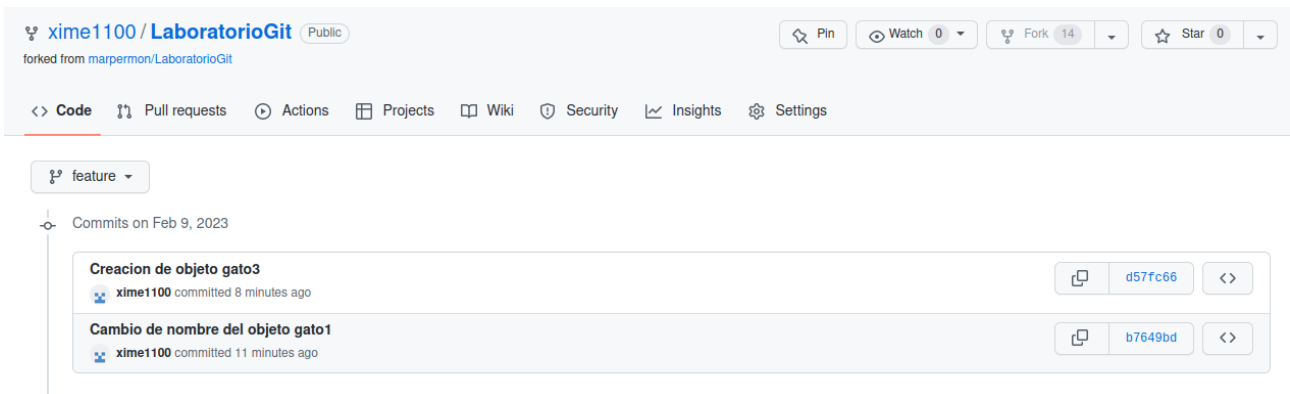


Figura 10: Comprobación de la rama feature en el repositorio remoto

3. Cambios en la rama main

Seguidamente, se utiliza el comando “git checkout” para trasladarse a la rama main, luego mediante el editor de texto nano, se realiza el cambio de nombre del gato 1, como se muestra

en la figura 11, seguido de esto se procede a guardar los cambios y realizar el commit, como se puede observar en la figura 12

```
ximenacq@ximena-vm: ~/LaboratorioGit
GNU nano 4.8 gatosGit/gatos.py
import funcionesTarea

gato1 = funcionesTarea.gato()
gato1.nombre = "Lia"
gato1.color = "negro"
gato1.edad = 4
gato1.aniadirlista()
```

Figura 11: Cambio de nombre gato1 desde main

```
ximenacq@ximena-vm:~/LaboratorioGit$ git log
commit 96d1ada14b8d7d99890a9087dc058d161723ed60 (HEAD -> main)
Author: xime1100 <ximena.cespedesquesada@ucr.ac.cr>
Date: Thu Feb 9 20:06:49 2023 -0600

    Cambio de nombre gato1 branch main

commit d57fc66e55ab07f6de3648fe4077a76069ca0702 (origin/feature, feature)
Author: xime1100 <ximena.cespedesquesada@ucr.ac.cr>
Date: Thu Feb 9 19:55:08 2023 -0600

    Creacion de objeto gato3

commit b7649bd200f64e8b598e329413cd26bbe2b19c73
Author: xime1100 <ximena.cespedesquesada@ucr.ac.cr>
Date: Thu Feb 9 19:52:30 2023 -0600

    Cambio de nombre del objeto gato1
```

Figura 12: Creación del tercer commit

4. Inclusión de la rama feature en main

Ahora bien, para incluir la rama feature en main, se puede hacer uso de dos comando Merge/rebase, en este caso se utiliza el comando “rebase”.

Para realizar esta inclusión de la rama feature, es necesario estar ubicados en la rama main, y seguidamente se utiliza el comando “git rebase”, al correr este comando, como era de esperarse, se muestra un conflicto entre ramas, tal y como se muestra en la figura 13,


```
ximenacq@ximena-vm:~/LaboratorioGit$ git rebase feature
First, rewinding head to replay your work on top of it...
Applying: Cambio de nombre gato1 branch main
Using index info to reconstruct a base tree...
M       gatosGit/gatos.py
Falling back to patching base and 3-way merge...
Auto-merging gatosGit/gatos.py
CONFLICT (content): Merge conflict in gatosGit/gatos.py
error: Failed to merge in the changes.
Patch failed at 0001 Cambio de nombre gato1 branch main
hint: Use 'git am --show-current-patch' to see the failed patch
Resolve all conflicts manually, mark them as resolved with
"git add/rm <conflicted_files>", then run "git rebase --continue".
You can instead skip this commit: run "git rebase --skip".
To abort and get back to the state before "git rebase", run "git rebase --abort".
```

Figura 13: Inclusión de feature en main

5. Conflictos

Ahora bien, al ingresar al editor de texto nano, se pueden observar mas detallado los conflictos que existen en las dos ramas, en este caso seria en la variable gato1, como se observa en la figura 14

```
ximenacq@ximena-vm: ~/LaboratorioGit
GNU nano 4.8                                gatosGit/gatos.py
import funcionesTarea

gato1 = funcionesTarea.gato()
<<<<<<< HEAD
gato1.nombre = "Lia"
=====
gato1.nombre = "Campanita"
>>>>>>> feature
gato1.color = "negro"
gato1.edad = 4
gato1.aniadirlista()

gato2 = funcionesTarea.gato()
gato2.nombre = "Pelusa"
gato2.color = "cafe"
gato2.edad = 3
gato2.aniadirlista()

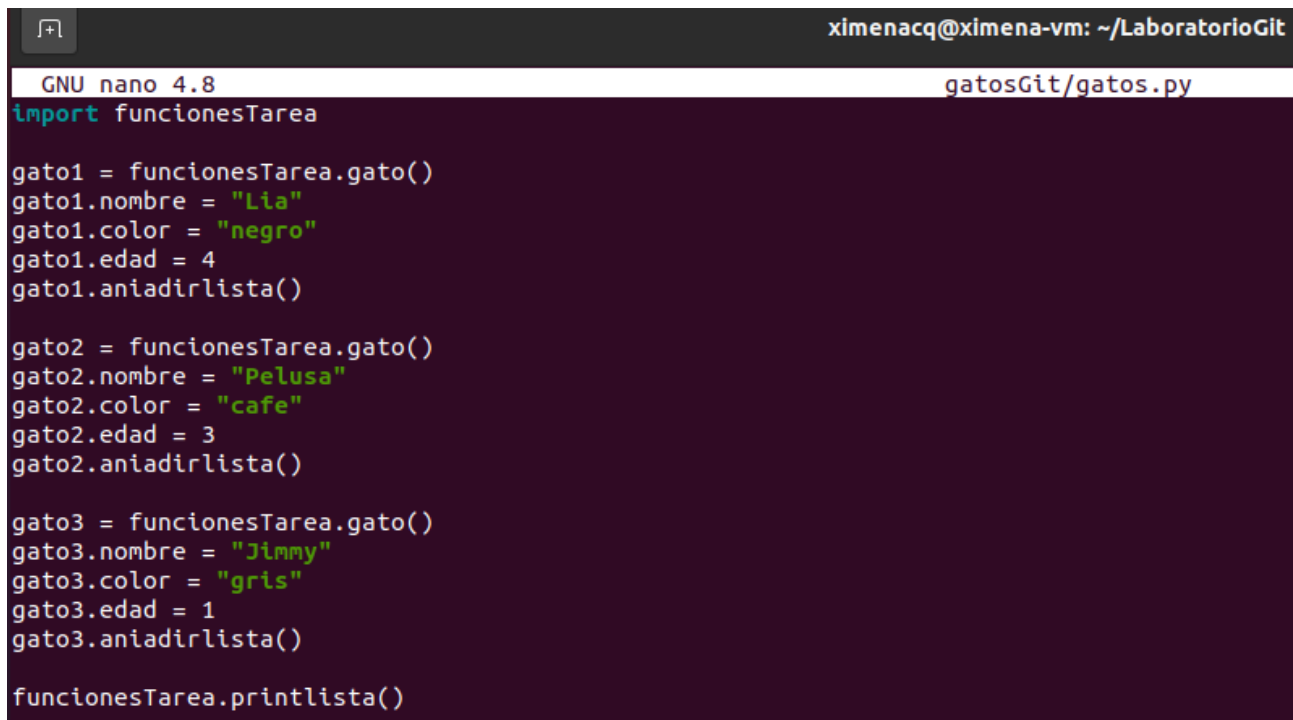
gato3 = funcionesTarea.gato()
gato3.nombre = "Jimmy"
gato3.color = "gris"
gato3.edad = 1
gato3.aniadirlista()

funcionesTarea.printlista()
```

Figura 14: Conflicto entre ramas

6. Resolución de conflictos

Ahora bien, para resolver el conflicto se procede a dejar la variable gato1, como se había cambiado en main, tal y como se observa en la figura 15.



```
ximenacq@ximena-vm: ~/LaboratorioGit
GNU nano 4.8                                gatosGit/gatos.py
import funcionesTarea

gato1 = funcionesTarea.gato()
gato1.nombre = "Lia"
gato1.color = "negro"
gato1.edad = 4
gato1.aniadirlista()

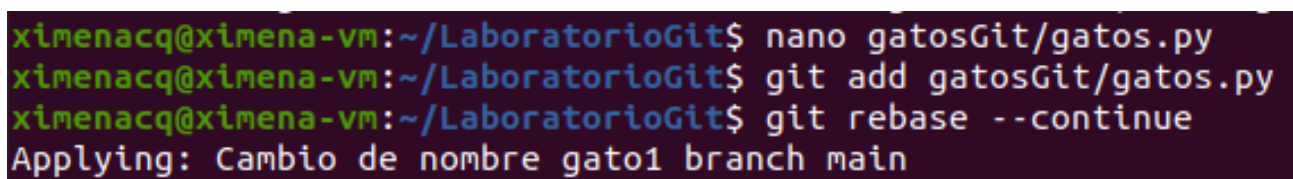
gato2 = funcionesTarea.gato()
gato2.nombre = "Pelusa"
gato2.color = "cafe"
gato2.edad = 3
gato2.aniadirlista()

gato3 = funcionesTarea.gato()
gato3.nombre = "Jimmy"
gato3.color = "gris"
gato3.edad = 1
gato3.aniadirlista()

funcionesTarea.printlista()
```

Figura 15: Solución del conflicto

Seguidamente, para aplicar la solución se procede a utilizar el comando “git rabase – continue” el cual indica que continúe con el “rebase” que se corrió anteriormente, como se muestra en la figura 16



```
ximenacq@ximena-vm:~/LaboratorioGit$ nano gatosGit/gatos.py
ximenacq@ximena-vm:~/LaboratorioGit$ git add gatosGit/gatos.py
ximenacq@ximena-vm:~/LaboratorioGit$ git rebase --continue
Applying: Cambio de nombre gato1 branch main
```

Figura 16: Aplicación de la solución.

7. Cambios en el repositorio remoto

Como se observa en la figura 17. Para incluir los cambios en el repositorio remoto, primeramente se utiliza el comando “fetch” el cual “sincroniza” la rama main del repositorio remoto y la rama main de la computadora, en segundo lugar, se utiliza el comando “rebase”, el cual nos indica que la rama main se encuentra sincronizada. Por ultimo, mediante el comando “push” se le indica al programa que mueva todo lo que se tiene en la rama main local a la rama main en el repositorio remoto.

```
ximenacq@ximena-vm:~/LaboratorioGit$ git fetch origin main
From github.com:xime1100/LaboratorioGit
* branch          main          -> FETCH_HEAD
ximenacq@ximena-vm:~/LaboratorioGit$ git rebase origin/main
Current branch main is up to date.
ximenacq@ximena-vm:~/LaboratorioGit$ git push origin main:main
Warning: Permanently added the ECDSA host key for IP address '140.82.114.4' to the list of known hosts.
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 424 bytes | 424.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:xime1100/LaboratorioGit.git
1640128..96d1ada  main -> main
```

Figura 17: Cambios al repositorio remoto.

Seguidamente, en la figura 18 se observa como los cambios fueron añadidos exitosamente.

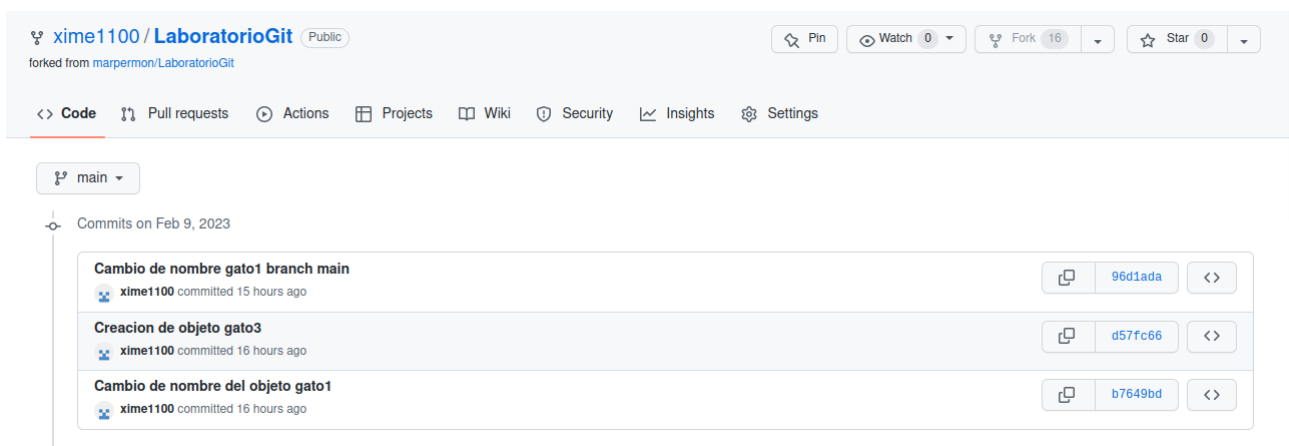


Figura 18: Verificación de que los cambios hayan sido incluidos al repositorio remoto.

Luego, como se observa en la figura 19, se corre el comando “git log –online –graph”, el cual muestra una linea del tiempo de los commits que se han realizado.

```
ximenacq@ximena-vm:~/LaboratorioGit$ git log --online --graph
* 96d1ada (HEAD -> main, origin/main, origin/HEAD) Cambio de nombre gato1 branch main
* d57fc66 (origin/feature, feature) Creacion de objeto gato3
* b7649bd Cambio de nombre del objeto gato1
* 1640128 included gato2 and list printing
* c2b877a corrected mistakes
* d02a6d2 included gato1
* 7ab0adc included functions file
* a13fd08 first commit
```

Figura 19: Comando “git log –online –graph” .

8. Manual git log

Al utilizar los comandos “git log –help” se ingresa al manual de ayuda para el comando “git log”.

- Graph:

```
--graph
Draw a text-based graphical representation of the commit history on the left hand side of the output. This may cause extra
lines to be printed in between commits, in order for the graph history to be drawn properly. Cannot be combined with --no-walk.

This enables parent rewriting, see History Simplification above.

This implies the --topo-order option by default, but the --date-order option may also be specified.
```

Figura 20: Comando “–graph” .

- Oneline:

```
--oneline
This is a shorthand for "--pretty=oneline --abbrev-commit" used together.
```

Figura 21: Comando “–oneline” .

- Diseño de Oneline:

```
• oneline

    <hash> <title line>

This is designed to be as compact as possible.
```

Figura 22: Para que fue diseñado “–oneline” .

9. Archivo gatos.py

Seguidamente, se corre el archivo gatos.py, como se muestra en la figura 23

```
ximenacq@ximena-vm:~/LaboratorioGit$ python3 gatosGit/gatos.py
Lia, 4 annos, color: negro
Pelusa, 3 annos, color: cafe
Jimmy, 1 annos, color: gris
```

Figura 23: Archivo gatos.py .

10. Repositorio local

Primeramente, se crea el pdf con las seis imágenes de los puntos 5, 7, 8 y 9. Seguidamente, como se muestra 24, se agrega el pdf con el comando “add”, para seguidamente crear el commit.

```
ximenacq@ximena-vm:~/LaboratorioGit$ git add ss.pdf
ximenacq@ximena-vm:~/LaboratorioGit$ git commit -m "ScreenShots puntos 5,7,8 y 9"
[main 5b77d15] ScreenShots puntos 5,7,8 y 9
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ss.pdf
ximenacq@ximena-vm:~/LaboratorioGit$ git log
commit 5b77d15c9025d7dc9bdc9b4d1ec3c99b9b4cc8 (HEAD -> main)
Author: xime1100 <ximena.cespedesquesada@ucr.ac.cr>
Date:   Fri Feb 10 15:23:42 2023 -0600

    ScreenShots puntos 5,7,8 y 9
```

Figura 24: Creación del pdf.

11. Push

Por ultimo, se sincronizan los datos de la rama main en el repositorio remoto a los datos que se tienen en el repositorio local, para seguidamente realizar un “push” de la rama main en el repositorio local al repositorio remoto, como se muestra en la figura 25.

```
ximenacq@ximena-vm:~/LaboratorioGit$ git fetch origin main
From github.com:xime1100/LaboratorioGit
 * branch          main          -> FETCH_HEAD
ximenacq@ximena-vm:~/LaboratorioGit$ git rebase origin/main
Current branch main is up to date.
ximenacq@ximena-vm:~/LaboratorioGit$ git push origin main:main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 82.50 KiB | 6.35 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:xime1100/LaboratorioGit.git
 96d1ada..5b77d15  main -> main
```

Figura 25: Push de la rama main al repositorio remoto.

Por ultimo, se verifica que los cambios se hayan realizado correctamente, como se observa en la figura 26, que el pdf se subió con éxito.

The screenshot shows the GitHub interface for a repository named 'xime1100 / LaboratorioGit'. The repository is public and was forked from 'marpermon/LaboratorioGit'. The main branch is 'main', and there are 2 branches and 0 tags. A notification indicates that the main branch is not protected. The repository is 4 commits ahead of the upstream 'marpermon:main'. The commit history shows three commits: 'gatosGit' (19 hours ago), 'README.md' (last month), and 'ss.pdf' (1 hour ago).

xime1100 / LaboratorioGit Public
forked from marpermon/LaboratorioGit

<> Code Pull requests Actions Projects Wiki Security Insights Settings

main 2 branches 0 tags Go to file Add file <> Code

Your main branch isn't protected
Protect this branch from force pushing or deletion, or require status checks before merging. [Learn more](#) Protect this branch

This branch is 4 commits ahead of marpermon:main. Contribute Sync fork

xime1100 ScreenShots puntos 5,7,8 y 9		5b77d15 1 hour ago 9 commits
gatosGit	Cambio de nombre gato1 branch main	19 hours ago
README.md	first commit	last month
ss.pdf	ScreenShots puntos 5,7,8 y 9	1 hour ago

Figura 26: Verificación de los cambios en el repositorio remoto.