

Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE-0117 Programación Bajo Plataformas Abiertas
III ciclo 2022

Laboratorio
Git/GitHub

Manfred Soza Garcia - B97755

Profesor: Julian Gairaud

19-02-2023

Índice

1. Indicaciones	1
1.1. Haga un fork del repositorio en su repositorio personal	1
1.2. Haga un merge de la rama feature hacia la rama main	1
1.3. Tome un screenshot de su editor de texto mostrando los conflictos de ambas ramas	1
1.4. Punto 7	1
1.5. Tome un screenshot de su terminal mostrando para qué sirve el comando <code>–graph</code>	2
1.6. Tome un screenshot de su terminal mostrando para qué sirve el comando <code>–online</code>	2
1.7. Tome un screenshot del archivo <code>gatos.py</code> resultante corriendo en su computadora	2

Índice de figuras

1.	Fork en repositorio personal.	1
2.	comando merge	1
3.	conflictos de ambas ramas	1
4.	Uso del comando git log --oneline --graph.	1
5.	Comando --graph	2
6.	Comando --oneline	2
7.	Archivo gatos.py en la terminal	2

1. Indicaciones

1.1. Haga un fork del repositorio en su repositorio personal

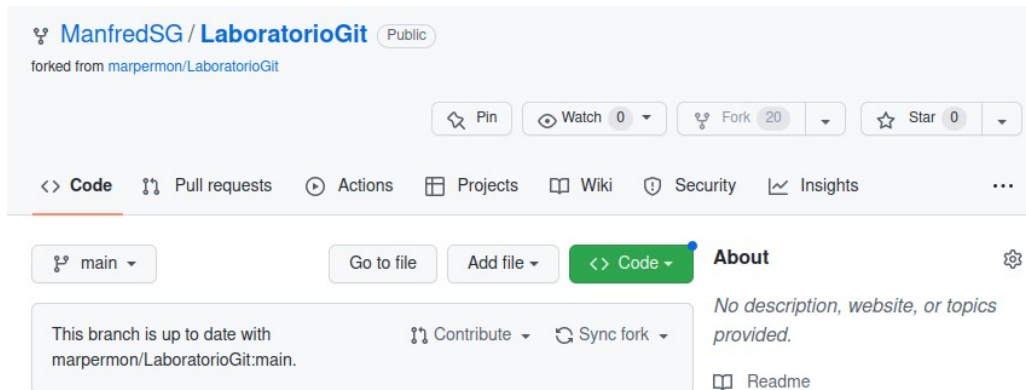


Figura 1: Fork en repositorio personal.

1.2. Haga un merge de la rama feature hacia la rama main

```
Updating 1640128..89cf4bf
error: Your local changes to the following files would be overwritten by merge:
      gatosGit/gatos.py
Please commit your changes or stash them before you merge.
Aborting
```

Figura 2: comando merge

1.3. Tome un screenshot de su editor de texto mostrando los conflictos de ambas ramas

```
Auto-merging gatosGit/gatos.py
CONFLICT (content): Merge conflict in gatosGit/gatos.py
Automatic merge failed; fix conflicts and then commit the result.
```

Figura 3: conflictos de ambas ramas

1.4. Punto 7

```
* 3750a20 (HEAD -> main, origin/main, origin/HEAD) Update gatos.py
* 1640128 included gato2 and list printing
* c2b877a corrected mistakes
* d02a6d2 included gato1
* 7ab0adc included functions file
* a13fd08 first commit
manfredsg@le0117:~/LaboratorioGit/gatosGit$
```

Figura 4: Uso del comando git log --oneline --graph.

- 1.5. Tome un screenshot de su terminal mostrando para qué sirve el comando `–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 5: Comando `–graph`

- 1.6. Tome un screenshot de su terminal mostrando para qué sirve el comando `–oneline`

```
PRETTY FORMATS
  If the commit is a merge, and if the pretty-format is not oneline,
  email or raw, an additional line is inserted before the Author: line.
  This line begins with "Merge: " and the hashes of ancestral commits are
  printed, separated by spaces. Note that the listed commits may not
  necessarily be the list of the direct parent commits if you have
  limited your view of history: for example, if you are only interested
  in changes related to a certain directory or file.

  There are several built-in formats, and you can define additional
  formats by setting a pretty.<name> config option to either another
  format name, or a format: string, as described below (see git-
  config(1)). Here are the details of the built-in formats:

  • oneline

    <hash> <title line>

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

Figura 6: Comando `–oneline`

- 1.7. Tome un screenshot del archivo `gatos.py` resultante corriendo en su computadora

```
manfredsg@ie0117:~/LaboratorioGit/gatosGit$ python3 gatos.py
Sussy, 4 annos, color: negro
Pelusa, 3 annos, color: cafe
Tanjiroo, 5 annos, color: Rojoo
manfredsg@ie0117:~/LaboratorioGit/gatosGit$
```

Figura 7: Archivo `gatos.py` en la terminal