



Universidad Tecnológica Nacional

# Programación I



Autor: Zullo, Lautaro

Tutor: Gubiotti, Flor

Profesor: AUS Bruselario, Sebastián

Marzo, 2025

# Índice

<b>Actividad 1 - Preguntas.....</b>	<b>2</b>
1.1. ¿Qué es GitHub?.....	3
1.2. ¿Cómo crear un repositorio en GitHub?.....	3
1.3. ¿Cómo crear una rama en Git?.....	3
1.4. ¿Cómo cambiar a una rama en Git?.....	3
1.5. ¿Cómo fusionar ramas en Git?.....	3
1.6. ¿Cómo crear un commit en Git?.....	3
1.7. ¿Cómo enviar un commit a GitHub?.....	4
1.8. ¿Qué es un repositorio remoto?.....	4
1.9. ¿Cómo agregar un repositorio remoto a Git?.....	4
1.10. ¿Cómo empujar cambios a un repositorio remoto?.....	4
1.11. ¿Cómo tirar de cambios de un repositorio remoto?.....	4
1.12. ¿Qué es un fork de repositorio?.....	4
1.13. ¿Cómo crear un fork de un repositorio?.....	4
1.14. ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?.....	5
1.15. ¿Cómo aceptar una solicitud de extracción?.....	5
1.16. ¿Qué es una etiqueta en Git? - ¿Cómo crear una etiqueta en Git? - ¿Cómo enviar una etiqueta a GitHub?.....	5
1.17. ¿Qué es un historial de Git? - ¿Cómo ver el historial de Git? - ¿Cómo buscar en el historial de Git? - ¿Cómo borrar el historial de Git?.....	6
1.18. ¿Qué es un repositorio privado en GitHub? - ¿Cómo crear un repositorio privado en GitHub? - ¿Cómo invitar a alguien a un repositorio privado en GitHub?.....	6
1.19. ¿Qué es un repositorio público en GitHub? - ¿Cómo crear un repositorio público en GitHub? - ¿Cómo compartir un repositorio público en GitHub?.....	6
<b>Actividad 2 - Github.....</b>	<b>8</b>
2. Actividades con github:.....	8
<b>Actividad 3 - Conflictos en github.....</b>	<b>9</b>
3. Más actividades con github:.....	9

## Actividad 1 - Preguntas

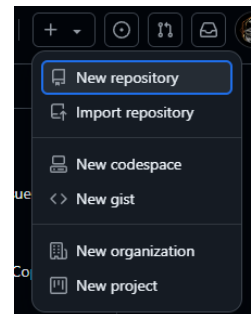
1. Contestar las siguientes preguntas utilizando las guías y documentación proporcionada

- 1.1. ¿Qué es **GitHub**?

Es una plataforma web basada en la nube donde los desarrolladores pueden subir, compartir y colaborar en repositorios.

- 1.2. ¿Cómo crear un repositorio en **GitHub**?

En la plataforma web vamos a encontrar un botón en la esquina superior derecha con un símbolo “+”, apretando ahí se va a desplegar un submenú con distintas opciones. Nosotros para crear un repositorio vamos a presionar donde dice “New repository” y de esta forma crearemos nuestro repositorio.



- 1.3. ¿Cómo crear una rama en **Git**?

Para crear una rama en **git** podemos ejecutar el siguiente comando “git branch <nombre\_rama>”, de esta forma se crea nuestra rama que podemos visualizar utilizando el comando “git branch”.

- 1.4. ¿Cómo cambiar a una rama en **Git**?

Para cambiar de rama en **git** tenemos que ejecutar el siguiente comando “git checkout <nombre\_rama>”.

- 1.5. ¿Cómo fusionar ramas en **Git**?

Para fusionar 2 ramas en **git** tendremos que ejecutar “git merge <nombre\_rama>”, esto fusiona la rama “nombre\_rama” con la rama en la que estamos parados actualmente.

- 1.6. ¿Cómo crear un commit en **Git**?

Para crear un commit en **git** solo tenemos que ejecutar el siguiente comando “git commit”, esto creará automáticamente un commit en nuestro repositorio. Si al commit le queremos agregar un mensaje tendremos que ponerle “-m” al comando quedando de esta forma “git commit -m “mensaje””

# Programación I

## 1.7. ¿Cómo enviar un commit a [GitHub](#)?

Para enviar nuestro commit a [github](#) lo que tenemos que hacer es subirlo a nuestro repositorio remoto creado en la plataforma. Esto se consigue con el siguiente comando “git push origin <nombre\_rama>”. La rama “<nombre\_rama>” es donde está alojado el commit que queremos subir.

## 1.8. ¿Qué es un repositorio remoto?

Es una copia de un proyecto que se encuentra alojada en un servidor remoto. Un ejemplo de esto es un repositorio de [github](#).

## 1.9. ¿Cómo agregar un repositorio remoto a [Git](#)?

Para añadir un repositorio remoto a [git](#) lo único que tenemos que hacer es ejecutar este comando “git remote add origin <url>” donde “<url>” es la url donde está alojado este repositorio.

## 1.10. ¿Cómo empujar cambios a un repositorio remoto?

La primera vez que empujemos cambios a un repositorio remoto vamos a tener que ejecutar el comando “git push -u origin <nombre\_rama>” donde “<nombre\_rama>” va a ser la rama comúnmente llamada “main / master” que nosotros definimos en nuestro repositorio.

## 1.11. ¿Cómo tirar de cambios de un repositorio remoto?

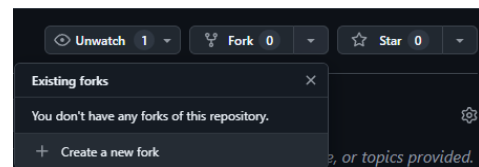
Para bajar los cambios de un repositorio remoto vamos a utilizar el comando “git pull”.

## 1.12. ¿Qué es un fork de repositorio?

Un fork de un repositorio es una copia exacta del repositorio original. Esto hicieron por ejemplo con linux para crear sus distintas distribuciones.

## 1.13. ¿Cómo crear un fork de un repositorio?

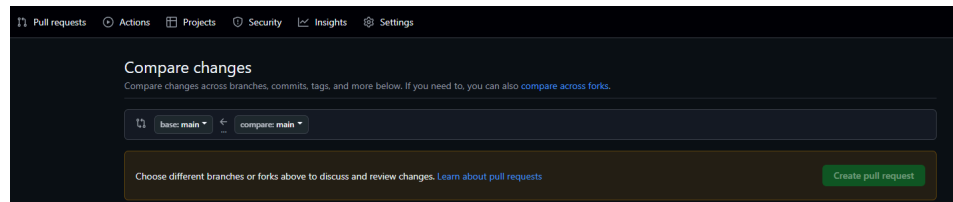
Para crear un fork en un repositorio tenemos que dirigirnos al repositorio en cuestión dentro de [github](#), en la esquina superior derecha encontraremos un botón que dice “Fork”, presionamos y elegimos la opción de “Create new fork” y de esta forma ya tendremos creado nuestro fork en [github](#).



# Programación I

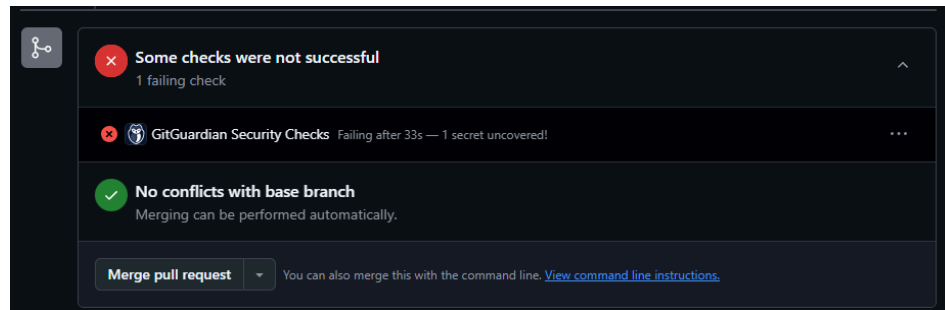
## 1.14. ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

Para crear un pull request podemos hacerlo desde [github](#). Una vez dentro del repositorio nos dirigimos al apartado “Pull request” que está en el submenú debajo del título del repositorio, apretamos el botón verde que dice “New pull request”. Dentro de este panel seleccionamos “<rama a la que va el cambio> ← <rama en la que está el cambio>” y presionamos “Create pull request”



## 1.15. ¿Cómo aceptar una solicitud de extracción?

Una vez creado nuestro pull request, podemos entrar a ver los detalles desde la pestaña “Pull request” seleccionando el nombre del pull request que queremos aceptar. Una vez dentro, podremos ver todos los detalles y si hay conflictos entre las ramas a mergear (si hay conflictos los tendremos que resolver antes de aceptar el pull request), una vez resueltos los conflictos podremos aceptar el PR apretando el botón que dice “Merge pull request”.



(Ignorar los conflictos de seguridad, es un repo viejo y tiene unas claves de Django que no deberían estar ahí 😞😞)

## 1.16. ¿Qué es una etiqueta en Git? - ¿Cómo crear una etiqueta en Git? - ¿Cómo enviar una etiqueta a GitHub?

Es una referencia que se aplica a un punto específico en el historial de un repositorio. Para crear una etiqueta con un mensaje podemos utilizar el siguiente comando “git tag -a <nombre> -m “mensaje””. Para enviar esta etiqueta a [github](#), lo único que tendremos que hacer es ejecutar un [git push](#).

# Programación I

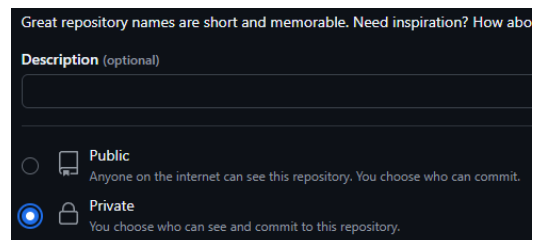
- 1.17. ¿Qué es un historial de **Git**? - ¿Cómo ver el historial de **Git**? - ¿Cómo buscar en el historial de **Git**? - ¿Cómo borrar el historial de **Git**?

El historial de **git** es un registro de los cambios realizados en un repositorio. Para ver el historial de **git** podemos utilizar el comando “git log” con esto obtendremos un listado de los commits que realizamos en nuestro repositorio. Con el comando “git grep” podemos buscar por medio de un regex coincidencias en nuestros commits. Para borrar el historial de commits de **git** se utiliza el comando “git rebase”.

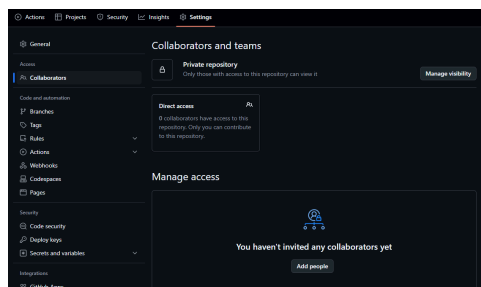
- 1.18. ¿Qué es un repositorio privado en **GitHub**? - ¿Cómo crear un repositorio privado en **GitHub**? - ¿Cómo invitar a alguien a un repositorio privado en **GitHub**?

Un repositorio es privado en **github** cuando la visibilidad del repositorio es limitada a los usuarios con acceso al mismo.

Para crear un repositorio privado tendremos que seleccionar esta opción al crear un repositorio. De esta forma cuando presionemos en el botón “Create repository” se nos habrá creado un repositorio

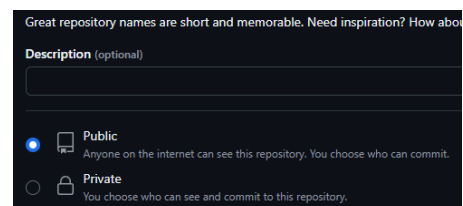


privado. Para invitar gente a nuestro repositorio privado tendremos que agregarlos como colaboradores desde el apartado de “settings → collaborators”.



- 1.19. ¿Qué es un repositorio público en **GitHub**? - ¿Cómo crear un repositorio público en **GitHub**? - ¿Cómo compartir un repositorio público en **GitHub**?

Un repositorio es público en **github** cuando la visibilidad del repositorio no está limitada, es decir, cualquier persona puede acceder al mismo. Para crear un repositorio público tendremos que seleccionar esta opción al crear un repositorio. De esta forma cuando presionemos en el botón “Create repository” se nos habrá creado un



## Programación I

repositorio público. Para compartir un repositorio público lo único que tendremos que hacer es copiar el link al repositorio y enviarlo. De esta forma podrán acceder al repositorio sin ningún tipo de problema.

## Actividad 2 - Github

### 2. Actividades con github:

#### 2.1. Crear un repositorio.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \*  /

Repository name \*

Great repository names are short and memorable. Need inspiration? [How about Rusty-githubs?](#)

Description (optional)

Public ☒ Anyone on the internet can see this repository. You choose who can commit.

Private ☐ Only those who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

Choose which files to track from a list of templates. [Learn more about creating files.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

☐ You are creating a public repository in your personal account.

[Create repository](#)

#### 2.2. Agregando un archivo “mi-archivo.txt”

```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  mi-archivo.txt

nothing added to commit but untracked files present (use "git add" to track)

$ git add .
$ git commit -m "Nuevo archivo txt."
[main 700c5cc] Nuevo archivo txt.
1 file changed, 1 insertion(+)
create mode 100644 mi-archivo.txt

$ git push origin main
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), 299 bytes | 299.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Lautiz/tp2_utm.git
d339564..700c5cc main -> main
```

#### 2.3. Crear una nueva rama y subir cambios

```
$ git checkout -b nueva-rama
Switched to a new branch 'nueva-rama'

$ git status
On branch nueva-rama
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  archivo-nueva-rama.txt

nothing added to commit but untracked files present (use "git add" to track)

$ git add .
$ git commit -m "Nuevo archivo nueva rama."
[nueva-rama f23451f] Nuevo archivo nueva rama.
1 file changed, 1 insertion(+)
create mode 100644 archivo-nueva-rama.txt

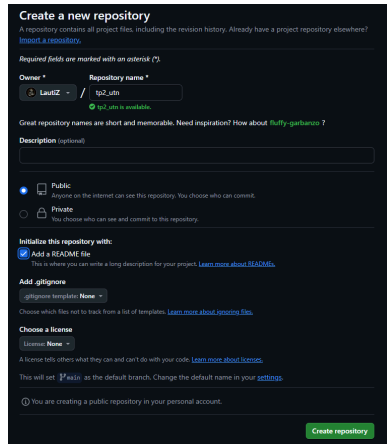
$ git push origin nueva-rama
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 358 bytes | 358.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
```



## Actividad 3 - Conflictos en github

### 3. Más actividades con github:

#### 3.1. Crear repositorio con README dentro



The screenshot shows the GitHub 'Create a new repository' page. It includes fields for 'Owner' (Lautiz) and 'Repository name' (tp2\_utm). There are options for 'Public' or 'Private' visibility. Under 'Initialize this repository with:', the 'Add a README file' option is selected. Other options include 'Add .gitignore' and 'Choose a license'. A 'Create repository' button is at the bottom right.

#### 3.2. Clonar repositorio a máquina local

```
MINGW64/d/Facultad/1. Semestre/Programacion1/TPS/tp2_utm
zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS
$ git clone https://github.com/Lautiz/tp2_utm.git
Cloning into 'tp2_utm'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 0), reused 6 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS
$ cd tp2_utm/

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utm (main)
$
```

#### 3.3. Crear nueva rama y editar archivo

```
MINGW64/d/Facultad/1. Semestre/Programacion1/TPS/tp2_utm
zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utm (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utm (feature-branch)
$ code .

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utm (feature-branch)
$ git status
On branch feature-branch
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:   README.md

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

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utm (feature-branch)
$ git add .

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utm (feature-branch)
$ git commit -m "Added a line"
[feature-branch eb56b48] Added a line
1 file changed, 3 insertions(+), 1 deletion(-)

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utm (feature-branch)
$
```

# Programación I

## 3.4. Volver a rama principal y editar el mismo archivo

```
MINGW64/d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn
zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn (main)
$ 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:   README.md

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

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn (main)
$ git add .

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn (main)
$ git commit -m "Added a line in main branch"
[main 823b2f0] Added a line in main branch
1 file changed, 3 insertions(+), 1 deletion(-)

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn (main)
$
```

## 3.5. Hacer merge y generar conflicto

```
MINGW64/d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn
zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn (main)
$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn (main|MERGING)
$ |
```

## 3.6. Resolver conflicto

```
MINGW64/d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn
zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn (main|MERGING)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

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

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

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

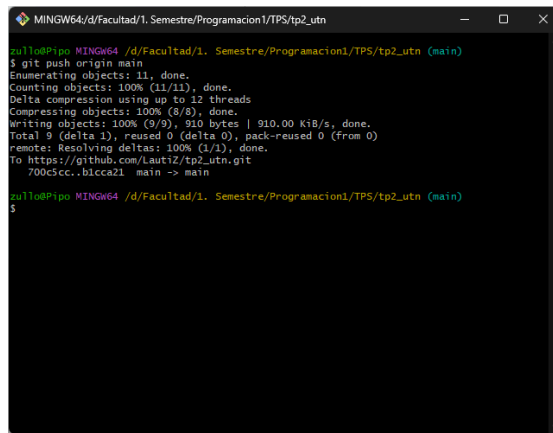
zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn (main|MERGING)
$ git add .

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main b1cca21] Resolved merge conflict

zullo@Pipo MINGW64 /d/Facultad/1. Semestre/Programacion1/TPS/tp2_utn (main)
$ |
```

# Programación I

## 3.7. Subir cambios a github

A terminal window titled 'MINGW64:/d/Facultad/1. Semestre/Programacion1/TP5/tp2\_utn' showing the execution of 'git push origin main'. The output shows the process of enumerating, counting, compressing, and writing objects, followed by resolving deltas and pushing to the remote repository at 'https://github.com/LautiZ/tp2\_utn.git'. The push is successful, updating the 'main' branch.

```
MINGW64:/d/Facultad/1. Semestre/Programacion1/TP5/tp2_utn (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 910 bytes | 910.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/LautiZ/tp2_utn.git
  700c5cc..b1cca21  main -> main
$
```

## 3.8. Verificar los cambios en github

