

## TP2 – Programación 1

**Alumno:** Lucas Couchot

1)

- ¿Qué es GitHub?

Es una plataforma en la nube que permite administrar repositorios GIT de forma remota. Posee varias características extra como la posibilidad de crear organizaciones, realizar forks, etc.

- ¿Cómo crear un repositorio en GitHub?

Para poder crear un repositorio en GitHub deberemos estar registrados si es que aún no lo hemos hecho. Una vez registrados deberemos iniciar sesión e ingresar a la opción “New repository” que nos va a llevar al formulario para dar de alta el nuevo repositorio indicando su nombre, además de otros parámetros.

Una vez creado el repositorio ya podemos clonarlo o bien configurar nuestro repositorio local para que apunte al generado en GitHub.

- ¿Cómo crear una rama en Git?

Para crear una rama puede usarse el comando “git branch <rama>” o bien “git checkout -b <rama>”

(La segunda opción nos posiciona en la nueva rama luego de crearla)

- ¿Cómo cambiar a una rama en Git?

Para cambiar de rama podemos usar el comando “git checkout <rama>”, esto nos posicionará en la rama deseada.

- ¿Cómo fusionar ramas en Git?

La forma más común de fusionar ramas en Git es utilizar el comando “git merge <rama\_a\_fusionar>”, lo cual intentará unir el historial de la rama indicada en la rama actual.

- ¿Cómo crear un commit en Git?

## TP2 – Programación 1

**Alumno:** Lucas Couchot

Para crear un commit en Git tenemos que usar el comando “git commit”, el commit se creará con los cambios que tengamos agregados al área de staging (esto usando se tuvo que haber realizado previamente con el comando “git add”).

- ¿Cómo enviar un commit a GitHub?

Si tenemos correctamente configurado el repositorio remoto, podemos enviar un commit a GitHub simplemente usando el comando “git push origin <repositorio\_remoto>”.

- ¿Qué es un repositorio remoto?

Es un repositorio pero que se encuentra en otro equipo diferente al mío y que hemos configurado localmente como tal. Normalmente le llamamos así al repositorio que existe en GitHub o alguna otra plataforma para almacenar repositorios.

- ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remote a git podemos usar el comando “git remote add”. Por ejemplo “git remote add origin <https://github.com/sbruselario/UTN-TUPaD-P1>”.

En este caso agregamos el repositorio con el nombre origin que suele ser la convención, cabe destacar que podríamos tener múltiples repositorios remotos aunque suele usarse sólo uno.

- ¿Cómo empujar cambios a un repositorio remoto?

De la misma forma que indicamos previamente, podemos subir cambios a un repositorio remoto con el comando “git push origin <repositorio\_remoto>”.

- ¿Cómo tirar de cambios de un repositorio remoto?

(Asumo que “tirar de” se refiere a “pull”)

Para sincronizar los cambios de un repositorio remoto con nuestro repositorio local podemos usar el comando “git pull”. Esto va a traerse los cambios del

## TP2 – Programación 1

**Alumno:** Lucas Couchot

repositorio remoto e intentará sincronizarlos con los del repositorio local. Es importante entender que esto puede intentar realizar un merge / rebase en caso de que las ramas hayan divergido.

- ¿Qué es un fork de repositorio?

El término fork en el contexto de git refiere a cuando copiamos un repositorio para poder incorporar cambios. Estos cambios podríamos llegar a proponerlos para que se incorporen al repositorio original o bien podemos mantener un repositorio paralelo que evoluciona de forma independiente del original.

Cabe aclarar que fork no es un comando válido en git, sin embargo las diferentes plataformas incorporan este concepto para simplificar el trabajo en proyectos de código abierto.

Por ejemplo, en GitHub tenemos la opción de fork que nos copiará un repositorio de un tercero en nuestra cuenta para que podamos trabajar con una copia del mismo. Además de esto posee la funcionalidad de simplificar la posibilidad de proponer nuestros cambios hacia el repositorio original a través de un “pull request” (También llamado “merge request” en otras plataformas).

- ¿Cómo crear un fork de un repositorio?

Crear un fork de un repositorio va a depender de la plataforma que estemos utilizando. En el caso de GitHub tenemos un botón con la opción “Fork” cuando visitamos un repositorio que no nos pertenece.

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

Generar un “pull request” depende de cada plataforma (como es el caso de realizar un fork). En el caso de GitHub nos va a aparecer la opción una vez tengamos creado el fork en la pantalla principal de nuestra copia del repositorio dentro del menú “Contribute”.

Luego de generar el “pull request”, el propietario (u otro usuario con los permisos necesarios) podrá revisar el mismo e incorporar nuestros cambios o descartarlos.

- ¿Cómo aceptar una solicitud de extracción?

## TP2 – Programación 1

**Alumno:** Lucas Couchot

Para aceptar un “pull request” en el caso de GitHub se puede utilizar el botón “merge” que aparecerá en el detalle del mismo. También podría realizarse esta tarea de forma local pero es un poco más complejo ya que implicaría descargarse el repositorio con los cambios localmente.

- ¿Qué es un etiqueta en Git?

Una etiqueta o tag no es más que un puntero a un commit en particular y nos sirve para poder marcar commits importantes en el repositorio. Por ejemplo podríamos usar un tag para indicar la publicación de una versión en particular.

- ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta en Git podemos usar el comando “git tag <tag>”. Por ejemplo para crear un tag “ver1.0.1” podemos usar el comando “git tag ver1.0.1”.

También podríamos generar un tag anotado al cual podemos incorporarle una descripción más detallada si usamos la variante “git tag -a <tag> -m <descripción>”.

- ¿Cómo enviar una etiqueta a GitHub?

Por defecto los tags sólo quedan localmente por lo que si queremos subirlos a GitHub u otro repositorio remoto deberemos usar el comando “git push origin <tag>” o bien, si queremos subir múltiples tags, “git push origin –tags”.

- ¿Qué es un historial de Git?

Historial en Git se refiere a la lista cronológica de commits de un repositorio.

- ¿Cómo ver el historial de Git?

Podemos visualizar el historial de Git para la rama actual con el comando “git log”. Otra opción bastante usada es el comando “git reflog”.

Ambas nos listan los commits de la rama ordenados de forma cronológica y nos dan detalles como el autor del mismo, la fecha del commit, etc.

## TP2 – Programación 1

**Alumno:** Lucas Couchot

- ¿Cómo buscar en el historial de Git?

Si necesitamos buscar en el historial de Git tenemos varias alternativas, una muy común es utilizar el buscador de GitHub o la plataforma que estemos usando para simplificar el trabajo.

Si necesitamos realizar una búsqueda en un repositorio de forma local existen varias opciones.

Una opción bastante sencilla es imprimir el historial a un archivo de texto y buscar con un editor de texto en el archivo resultante (`git --no-pager log > log.txt`).

Otra opción podría ser utilizar una expresión regular al ejecutar el comando log: `git log -grep=<regex>`.

- ¿Cómo borrar el historial de Git?

Para eliminar historial de Git normalmente se utiliza el comando `git rebase` que nos permite reescribir el historial del repositorio. Es un comando bastante complejo ya que nos permite eliminar o modificar los commits de forma individual, incluso pudiendo alterar el autor de los mismos.

Cabe destacar que reescribir el historial de un repositorio nos va a generar que la rama local difiera con la rama remota (si es que existe esta última), por lo que vamos a necesitar forzar nuestros cambios con el comando `git push -force` para que se vean reflejados en el repositorio remoto. Esto podría traer conflictos si otras personas tienen clonada la rama afectada en su equipo de forma local ya que van a notar una divergencia entre las ramas.

- ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub se refiere a un repositorio que no es visible públicamente por cualquier usuario de GitHub sino que lo pueden ver sólo los usuarios con los permisos correspondientes.

Es una opción bastante usada cuando estamos trabajando en proyectos que no deben o no pueden ser de código abierto.

- ¿Cómo crear un repositorio privado en GitHub?

## TP2 – Programación 1

**Alumno:** Lucas Couchot

Los repositorios privados en GitHub se generan de la misma forma que los públicos, sólo hay que tener el cuidado de seleccionar que es privado en el formulario de creación.

- ¿Cómo invitar a alguien a un repositorio privado en GitHub?

En el caso de GitHub podemos hacer uso de la opción “Collaborators” del repositorio para incorporar usuarios que puedan trabajar en el mismo.

Los usuarios podrán aceptar o no la invitación desde GitHub.

También podemos limitar qué acciones pueden realizar los colaboradores sobre el repositorio.

- ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub se trata de un repositorio que se encuentra visible para todo el mundo. Esto significa que todo el código o archivos del mismo son de acceso público.

Que un repositorio sea público no quiere decir que cualquier persona pueda modificar nuestros archivos, sólo se refiere a la visibilidad.

- ¿Cómo crear un repositorio público en GitHub?

De la misma forma que un repositorio privado, un repositorio público puede ser generado en GitHub a través de la opción para crear nuevos repositorios.

- ¿Cómo compartir un repositorio público en GitHub?

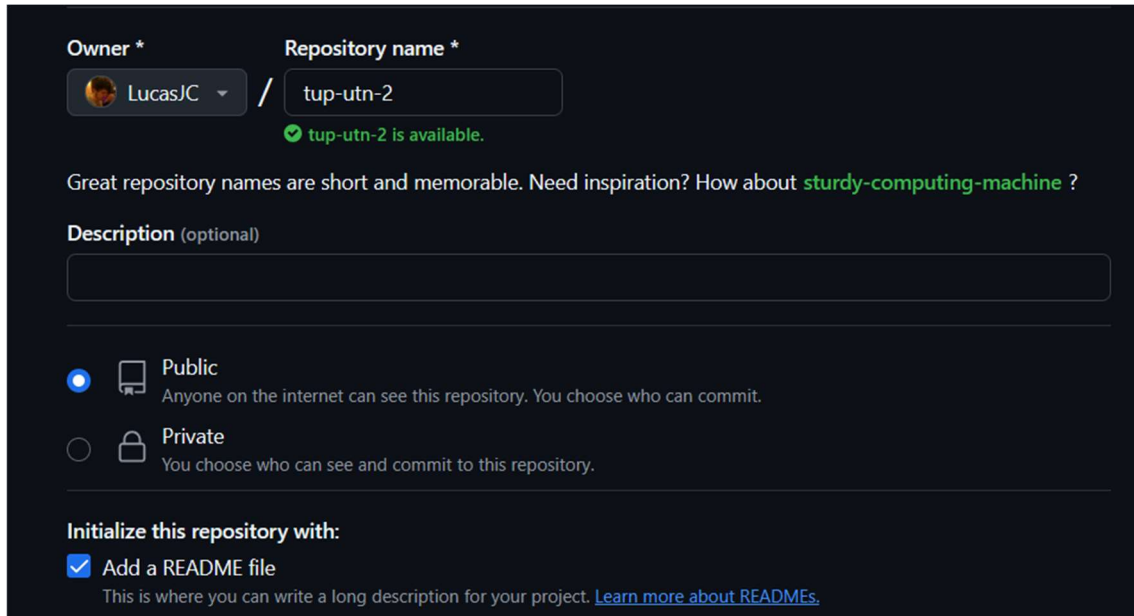
Podemos compartir un repositorio público almacenado en GitHub mediante su URL. Cada repositorio de GitHub posee una URL que lo identifica de forma única y puede ser usada para compartir. Cabe destacar que en caso de que nuestro repositorio sea privado, debemos asignar los permisos adecuados a los usuarios con los que deseemos compartir nuestro repositorio, de lo contrario no podrán ver el contenido del mismo.

## TP2 – Programación 1

**Alumno:** Lucas Couchot

2)

Creación del repositorio:



The screenshot shows the GitHub repository creation page. At the top, there are two input fields: 'Owner \*' with a dropdown menu showing 'LucasJC' and a profile picture, and 'Repository name \*' with the text 'tup-utn-2'. Below the repository name field, a green checkmark indicates 'tup-utn-2 is available.' Below this, there is a line of text: 'Great repository names are short and memorable. Need inspiration? How about **sturdy-computing-machine** ?'. Underneath is a 'Description (optional)' text area. Further down, there are two radio button options: 'Public' (selected) and 'Private'. The 'Public' option has a description: 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option has a description: 'You choose who can see and commit to this repository.' At the bottom, there is a section 'Initialize this repository with:' with a checked checkbox for 'Add a README file'. Below this checkbox, there is a note: 'This is where you can write a long description for your project. [Learn more about READMEs.](#)'

Clonamos el repositorio:

```
C:\Users\lucas\OneDrive\Documents\temp>git clone https://github.com/LucasJC/tup-utn-2.git
Cloning into 'tup-utn-2'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

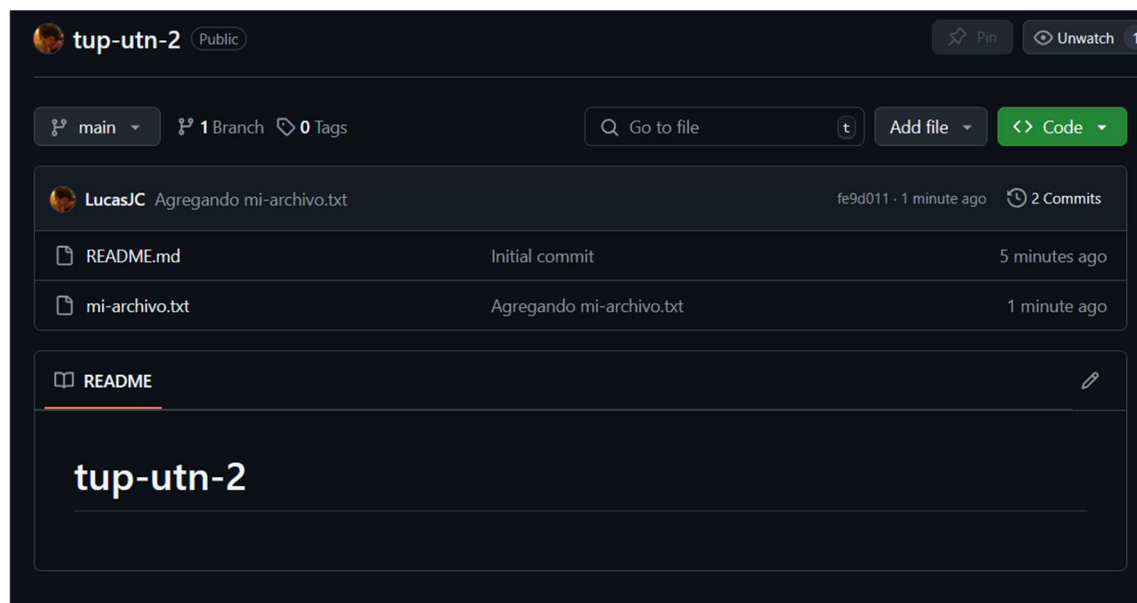
Creamos y subimos el archivo txt:

## TP2 – Programación 1

Alumno: Lucas Couchot

```
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>echo ejemplo > mi-archivo.txt
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git add mi-archivo.txt
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git commit -m "Agregando mi-archivo.txt"
[main fe9d011] Agregando mi-archivo.txt
1 file changed, 1 insertion(+)
create mode 100644 mi-archivo.txt
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 303 bytes | 303.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/LucasJC/tup-utn-2.git
10826bd..fe9d011 main -> main
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>dir
El volumen de la unidad C es Windows-SSD
El número de serie del volumen es: 14CE-727A

Directorio de C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2
26/03/2025 16:36 <DIR> .
26/03/2025 16:36 <DIR> ..
26/03/2025 16:36 10 mi-archivo.txt
26/03/2025 16:33 11 README.md
                2 archivos                21 bytes
                2 dirs 118.018.867.200 bytes libres
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>|
```



Creamos una nueva rama y subimos cambios al archivo txt:



## TP2 – Programación 1

Alumno: Lucas Couchot

```
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>echo ejemplo-modificado > mi-archivo.txt
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>type mi-archivo.txt
ejemplo-modificado
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>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:   mi-archivo.txt

no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git branch mis-cambios
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git checkout mis-cambios
Switched to branch 'mis-cambios'
M       mi-archivo.txt
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git add .
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git commit -m "Agrego cambios a mi-archivo.txt"
[mis-cambios ef10edf] Agrego cambios a mi-archivo.txt
1 file changed, 1 insertion(+), 1 deletion(-)
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git push -u origin mis-cambios
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 320 bytes | 320.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'mis-cambios' on GitHub by visiting:
remote:   https://github.com/LucasJC/tup-utn-2/pull/new/mis-cambios
remote:
To https://github.com/LucasJC/tup-utn-2.git
```

Branches			
Overview   Yours   Active   Stale   All			
Q Search branches...			
Default			
Branch	Updated	Check status	Behind   Ahead
<code>main</code>	5 minutes ago		<code>Default</code>
Your branches			
Branch	Updated	Check status	Behind   Ahead
<code>mis-cambios</code>	1 minute ago		0   1

3)

Comentario: Para simplificar vamos a utilizar el repositorio ya generado en el ejercicio 2).

Creamos una nueva rama y modificamos el README.md:

## TP2 – Programación 1

**Alumno:** Lucas Couchot

```
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git checkout -b conflicto
Switched to a new branch 'conflicto'

C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>echo texto-conflictivo > README.md

C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>type README.md
texto-conflictivo
```

```
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git add .

C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git commit -m "Modifico texto de README.md"
[conflicto 2c1f31a] Modifico texto de README.md
1 file changed, 1 insertion(+), 1 deletion(-)
```

Nos paramos en la rama main y editamos el mismo archivo, generando un conflicto:

```
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>echo otro-texto-diferente > README.md

C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git add .

C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git commit -m "Modifico texto de README.md desde main"
[main e5d943f] Modifico texto de README.md desde main
1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git merge conflicto
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

```
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>type README.md
<<<<<< HEAD
otro-texto-diferente
=====
texto-conflictivo
>>>>>> conflicto
```

Resolvemos manualmente los conflictos y generamos un nuevo commit para completar el merge. Finalmente subimos nuestros cambios al repositorio remoto:

## TP2 – Programación 1

Alumno: Lucas Couchot

```
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>type README.md
otro-texto-diferente texto-conflictivo
C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git add .

C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git commit -m "Resuelvo conflictos"
[main 4d8da3b] Resuelvo conflictos

C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 841 bytes | 841.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/LucasJC/tup-utn-2.git
   fe9d011..4d8da3b  main -> main

C:\Users\lucas\OneDrive\Documents\temp\tup-utn-2>
```

En GitHub podemos verificar que nuestros cambios se encuentran subidos correctamente:

