

Trabajo Práctico N°2 : Git & GitHub

1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada. (Desarrollar las respuestas) :

- **¿Qué es GitHub?**

Github es una plataforma de desarrollo colaborativo que se utiliza para alojar proyectos, mediante el uso de control de versiones Git.

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

Para crear un repositorio en GitHub, es necesario primero crear una cuenta en la plataforma. Una vez registrado y logueado, dentro del listado de repositorios, se hace click en el botón de new para crear uno nuevo. Esto nos lleva a un formulario el cual solicita varios datos de nuestro repositorio siendo su nombre, la privacidad del mismo y distintas configuraciones.

- **¿Cómo crear una rama en Git?**

Para crear una rama en Git es necesario escribir por consola el comando:
`git branch nombre-de-mi-rama`

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

Para cambiar a una rama se escribe el siguiente comando:
`git checkout nombre-rama`

- **¿Cómo fusionar ramas en Git?**

Para fusionar ramas en Git, es necesario estar en la rama principal a la que se le quiere añadir o fusionar la otra rama y se escribe el siguiente comando:
`git merge nombre-rama`

- **¿Cómo crear un commit en Git?**

Para crear un commit es necesario escribir el siguiente comando:
`git commit -m "mensaje de mi commit"`

- **¿Cómo enviar un commit a GitHub?**

Para enviar un cambio al repositorio remoto se utiliza el siguiente comando:
`git push -u origin master`

- **¿Qué es un repositorio remoto?**

Un repositorio remoto es un espacio virtual o un espacio en la nube en donde un desarrollador puede tener sus proyectos alojados y acceder a los mismos desde cualquier dispositivo.

- **¿Cómo agregar un repositorio remoto a Git?**

Para agregar un repositorio remoto a git se utiliza el siguiente comando:
`git remote add <nombre> <URL_del_repositorio>`

- **¿Cómo empujar cambios a un repositorio remoto?**

Para empujar cambios a un repositorio remoto se utiliza el siguiente comando:

`git push -u origin master`

- **¿Cómo tirar de cambios de un repositorio remoto?**

Para traer cambios de un repositorio local se utiliza el comando pull:

`git pull origin master`

- **¿Qué es un fork de repositorio?**

Un fork o bifurcación de un repositorio es la ramificación de un proyecto existente en otro repositorio, el cual copio a mi repositorio personal y puedo realizar modificaciones sin alterar el repositorio original.

- **¿Cómo crear un fork de un repositorio?**

Dentro del repositorio del proyecto, existe un botón que dice fork, el cual al hacer click nos permite generar la ramificación del repositorio dentro del nuestro.

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

Para enviar una solicitud de pull request a un repositorio, primero se debe hacer un fork del repositorio, realizar todos los cambios que sean necesarios y una vez pusheados, desde nuestro repositorio utilizar la opción de “contribute”, el cual se despliega y da un botón de “open pull request”.

- **¿Cómo aceptar una solicitud de extracción?**

Dentro de la pestaña “Pull Requests” se encuentra la lista de solicitudes de cambios, que permite revisarlos, hacer retroalimentación mediante comentarios y aprobarlos.

- **¿Qué es una etiqueta en Git?**

Una etiqueta es una herramienta que permite categorizar cambios específicos y de ésta manera tener mejor seguimiento de los mismos.

- **¿Cómo crear una etiqueta en Git?**

Se crea con el comando tag:

`git tag nombre-tag.`

- **¿Cómo enviar una etiqueta a GitHub?**

Una vez creada la etiqueta, se envía a GitHub a con el siguiente comando:

`git push --tags.`

- **¿Qué es un historial de Git?**

El historial de Git es una herramienta mediante gráficos que permite ver los cambios realizados.

- **¿Cómo ver el historial de Git?**

Para ver el historial de Git, se utiliza el comando log.

- **¿Cómo buscar en el historial de Git?**

Para buscar en el historial de Git, al comando log se le debe agregar el parámetro deseado, por ejemplo `–author=“nombre-autor”` o `–grep=“palabra-clave”`.

- **¿Cómo borrar el historial de Git?**

Se puede borrar el historial de los últimos commits mediante el comando `git reset`.

- **¿Qué es un repositorio privado en GitHub?**

Un repositorio privado es aquel al que tiene acceso solo el creador y las cuentas que él autorice.

- **¿Cómo crear un repositorio privado en GitHub?**

Al momento de crear un repositorio, el mismo formulario de creación tiene la opción de privacidad.

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

Para invitar a un colaborador, hay que ir a la configuración del repositorio y luego al apartado de colaboradores. En el mismo, se puede administrar el acceso al repositorio.

- **¿Qué es un repositorio público en GitHub?**

Es un repositorio que permite el acceso y el fork de otros colaboradores.

- **¿Cómo crear un repositorio público en GitHub?**

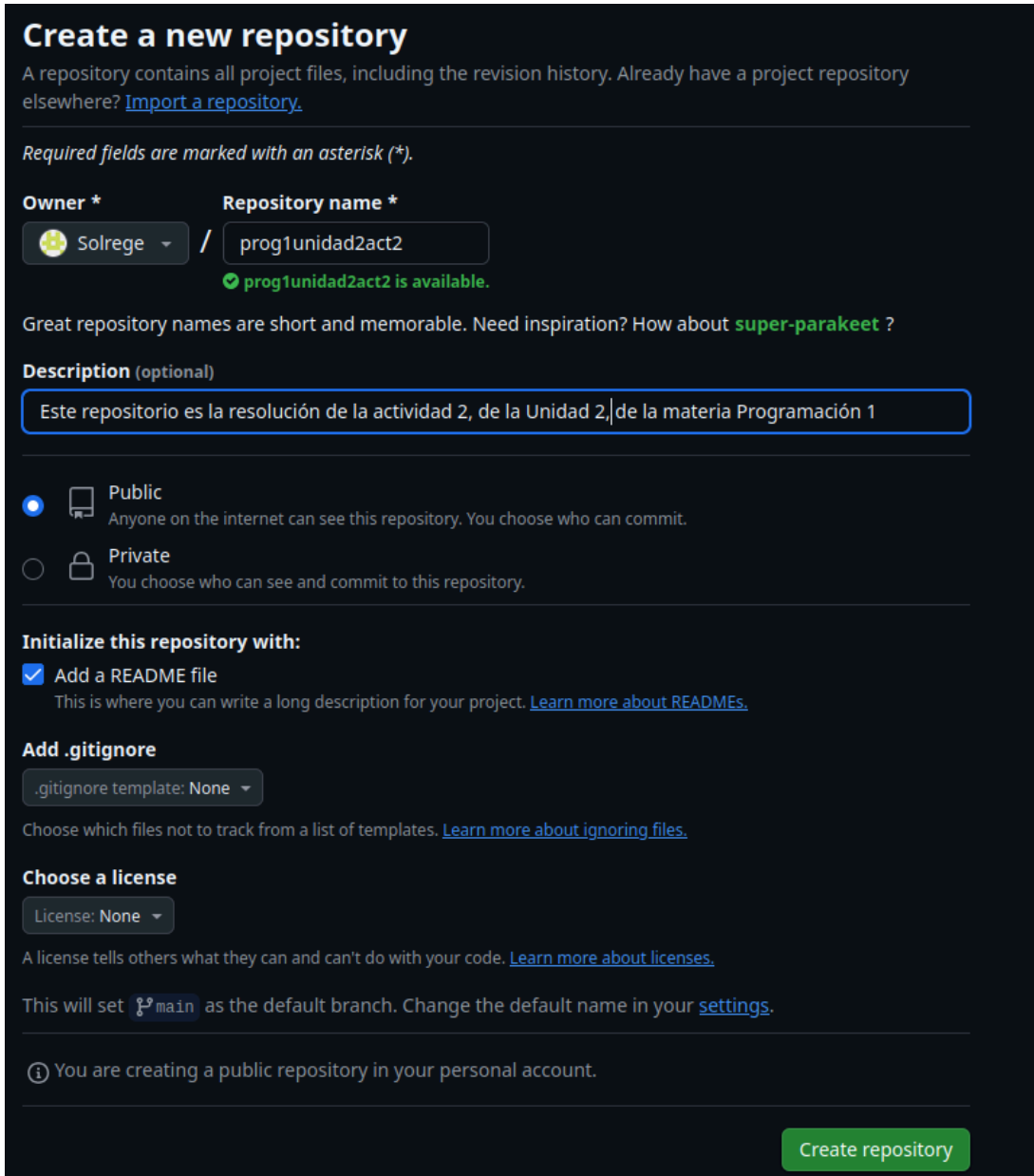
Al momento de crear un repositorio, el mismo formulario de creación tiene la opción de privacidad.

- **¿Cómo compartir un repositorio público en GitHub?**

Se puede compartir un repositorio público compartiendo la url del mismo.

2) Realizar la siguiente actividad:

- Crear un repositorio.
 - Dale un nombre al repositorio.
 - Elige que el repositorio sea público.
 - Inicializa el repositorio con un archivo.



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 * Solrege / **Repository name *** prog1unidad2act2
✓ prog1unidad2act2 is available.

Great repository names are short and memorable. Need inspiration? How about **super-parakeet** ?

Description (optional)
Este repositorio es la resolución de la actividad 2, de la Unidad 2, de la materia Programación 1

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

☐ **Private**
You choose 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
.gitignore template: **None** ▾
Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
License: **None** ▾
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](#)

- Agregando un Archivo
 - Crea un archivo simple, por ejemplo, "mi-archivo.txt".
 - Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.
 - Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).

```
~/Documents/utn/programacion1
(16:10:41)→ git clone git@github.com:Solrege/prog1unidad2act2.git
Cloning into 'prog1unidad2act2'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
~/Documents/utn/programacion1
(16:11:13)→ cd prog1unidad2act2
~/Documents/utn/programacion1/prog1unidad2act2
(16:11:25 on main)→ cat > mi-archivo.txt
Este es un ejemplo para la actividad 2, del Trabajo Práctica nro 2 de la materia Programación 1
~/Documents/utn/programacion1/prog1unidad2act2
(16:12:22 on main ★)→ git add .
~/Documents/utn/programacion1/prog1unidad2act2
(16:12:56 on main ✚)→ git commit -m "add mi-archivo.txt"
[main 43b0496] add mi-archivo.txt
1 file changed, 1 insertion(+)
create mode 100644 mi-archivo.txt
~/Documents/utn/programacion1/prog1unidad2act2
(16:13:12 on main)→ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 14 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 373 bytes | 373.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Solrege/prog1unidad2act2.git
04e74de..43b0496  main -> main
```

 Solrege add mi-archivo.txt	43b0496 · now	🕒 2 Commits
 README.md	Initial commit	4 minutes ago
 mi-archivo.txt	add mi-archivo.txt	now

prog1unidad2act2 / mi-archivo.txt

 **Solrege** add mi-archivo.txt 43b0496 · 2 minutes ago 🕒 History

Code Blame 1 lines (1 loc) · 97 Bytes 🧠 Code 55% faster with GitHub Copilot Raw 📄 📥 ✎ ⌵ ⌂

```
1 Este es un ejemplo para la actividad 2, del Trabajo Práctica nro 2 de la materia Programación 1
```

- Creando Branchs
 - Crear una Branch
 - Realizar cambios o agregar un archivo
 - Subir la Branch

```
(16:36:27 on main)→ git checkout -b my-branch
Switched to a new branch 'my-branch'
~/Documents/utn/programacion1/prog1unidad2act2
(16:36:31 on my-branch)→ cat >> mi-archivo.txt
Estoy agregando una modificación mediante una branch
~/Documents/utn/programacion1/prog1unidad2act2
(16:37:11 on my-branch *)→ git add .
~/Documents/utn/programacion1/prog1unidad2act2
(16:37:50 on my-branch *)→ git commit -m "add new branch"
[my-branch e6d7397] add new branch
1 file changed, 1 insertion(+), 1 deletion(-)
~/Documents/utn/programacion1/prog1unidad2act2
(16:38:06 on my-branch)→ git push origin my-branch
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 14 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 402 bytes | 402.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'my-branch' on GitHub by visiting:
remote:   https://github.com/Solrege/prog1unidad2act2/pull/new/my-branch
remote:
To github.com:Solrege/prog1unidad2act2.git
 * [new branch]      my-branch -> my-branch
```

Branches New branch

Overview **Yours** Active Stale All

Q Search branches...

Default

Branch	Updated	Check status	Behind	Ahead	Pull request
main	25 minutes ago			Default	

Your branches

Branch	Updated	Check status	Behind	Ahead	Pull request
my-branch	now		0	1	

my-branch 2 Branches 0 Tags Go to file Add file <> Code

This branch is 1 commit ahead of main. Contribute

Solrege add new branch e6d7397 · 2 minutes ago 3 Commits

README.md	Initial commit	30 minutes ago
mi-archivo.txt	add new branch	2 minutes ago

prog1unidad2act2 / mi-archivo.txt

Solrege add new branch e6d7397 · 2 minutes ago History

Code Blame 1 lines (1 loc) · 150 Bytes Code 55% faster with GitHub Copilot Raw Download Edit

```
1 Este es un ejemplo para la actividad 2, del Trabajo Práctica nro 2 de la materia Programación 1Estoy agregando una modificación mediante una branch
```

Link del repositorio: <https://github.com/Solrege/prog1unidad2act2>

3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub


- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz clic en "Create repository".

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 *

 Solrege ▾

Repository name *


prog1unidad2act3

 **prog1unidad2act3 is available.**


Great repository names are short and memorable. Need inspiration? How about **friendly-guacamole** ?

Description (optional)

Este repositorio es la resolución de la actividad 3, de la Unidad 2, de la materia Programación 1

☒  **Public**

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

☐  **Private**

You choose 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


.gitignore template: None ▾


Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

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

Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio.
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando.
- Entra en el directorio del repositorio:

```
(16:01:11)→ git clone git@github.com:Solrege/proglunidad2act3.git
—(Mon,Mar31)—
Cloning into 'proglunidad2act3'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
~/Documents/utn/programacion1
~/Documents/utn/programacion1
(sol@sol-dev:pts/1)
(16:49:09)→ cd proglunidad2act3
~/Documents/utn/programacion1/proglunidad2act3
(16:49:54 on main)→
```

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:

```
~/Documents/utn/programacion1/proglunidad2act3
(16:49:54 on main)→ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
```

- Abre el archivo README.md en un editor de texto y añade una línea nueva

<pre>1 # proglunidad2act3 2 Este repositorio es la resolución de la actividad 3, de la Un 3 4 Ahora estoy agregando un cambio mediante una branch 5</pre>	<h3>proglunidad2act3</h3> <p>Este repositorio es la resolución de la actividad 3, de la Unidad 2, de la materia Programación 1</p> <p>Ahora estoy agregando un cambio mediante una branch</p>
---	---

- Guarda los cambios y haz un commit

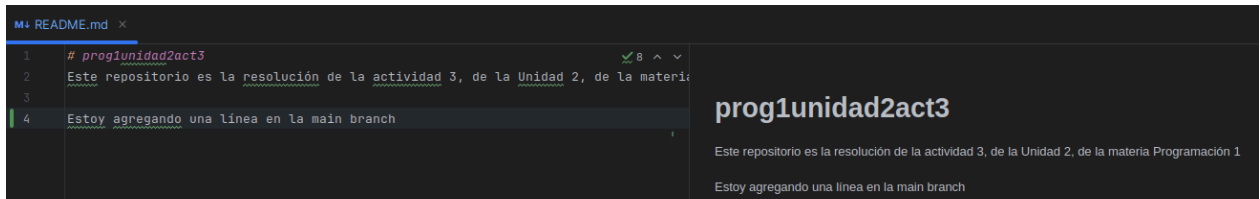
```
(16:59:53 on feature-branch ●)→ git add .
~/Documents/utn/programacion1/proglunidad2act3
(17:00:02 on feature-branch ●)→ git commit -m "Changed readme"
[feature-branch 5bde6d4] Changed readme
1 file changed, 2 insertions(+)
```


Paso 4: Volver a la rama principal y editar el mismo archivo

- Cambia de vuelta a la rama principal (main):

```
(17:00:49 on feature-branch) → git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

- Edita el archivo README.md de nuevo, añadiendo una línea diferente:



- Guarda los cambios y haz un commit:

```
(17:01:42 on main) → git add README.md
(~/.Documents/utn/programacion1/prog1unidad2act3)
(17:04:29 on main •) → git commit -m "Changed readme"
[main 7eda7d0] Changed readme
1 file changed, 2 insertions(+)
```

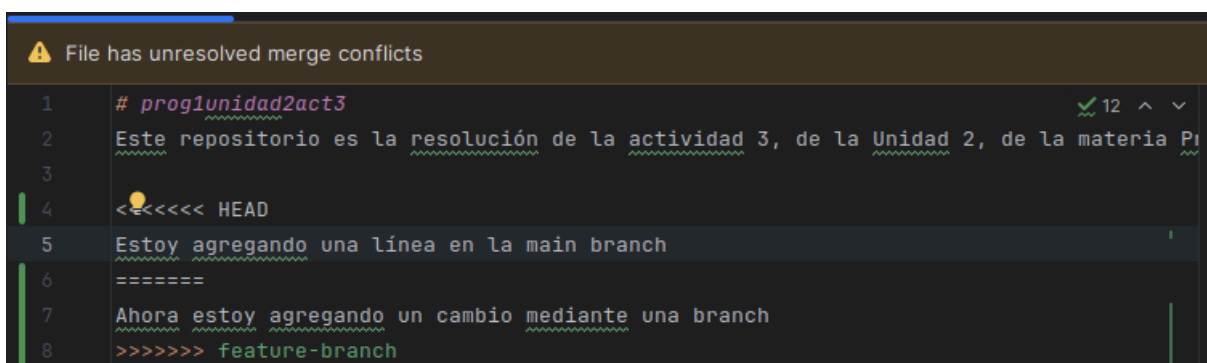
Paso 5: Hacer un merge y generar un conflicto

- Intenta hacer un merge de la feature-branch en la rama main:
- Se generará un conflicto porque los cambios afectan la misma línea.

```
(17:05:03 on 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.
```

Paso 6: Resolver el conflicto

- Abre el archivo README.md en tu editor de texto.



- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
- Edita el archivo para resolver el conflicto y guarda los cambios (Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).

```
1 # prog1unidad2act3
2 Este repositorio es la resolución de la actividad 3, de la Unidad 2, de la materia Pr
3
4 Estoy agregando una línea en la main branch
5
6 Ahora estoy agregando un cambio mediante una branch
7
8
```

- Añade el archivo resuelto y completa el merge:

```
(~/Documents/utn/programacion1/prog1unidad2act3)
(17:06:10 on main =>) git add README.md
(~/Documents/utn/programacion1/prog1unidad2act3)
(17:09:13 on main •) git commit -m "resolved merge commit"
[main 26f3230] resolved merge commit
```

Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub:

```
(~/Documents/utn/programacion1/prog1unidad2act3)
(17:09:30 on main) git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 14 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 787 bytes | 787.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To github.com:Solrege/prog1unidad2act3.git
18d7ae9..26f3230 main -> main
```

- También sube la feature-branch si deseas:

```
(17:10:24 on main) -> git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/Solrege/prog1unidad2act3/pull/new/feature-branch
remote:
To github.com:Solrege/prog1unidad2act3.git
* [new branch]      feature-branch -> feature-branch
```

Paso 8: Verificar en GitHub

- Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.



- Puedes revisar el historial de commits para ver el conflicto y su resolución

```
(~/Documents/utn/programacion1)
(15:04:00) -> git clone https://github.com/Solrege/UTN-Programacion1
Cloning into 'UTN-Programacion1'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (2/2), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 16 (delta 0), reused 0 (delta 0), pack-reused 14 (from 1)
Receiving objects: 100% (16/16), done.
```

Link del repositorio: <https://github.com/Solrege/prog1unidad2act3>