

TP 2: Trabajo Colaborativo

(Git y GitHub)

Alumno: Macarena Cantoni (Mat: 100263)

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

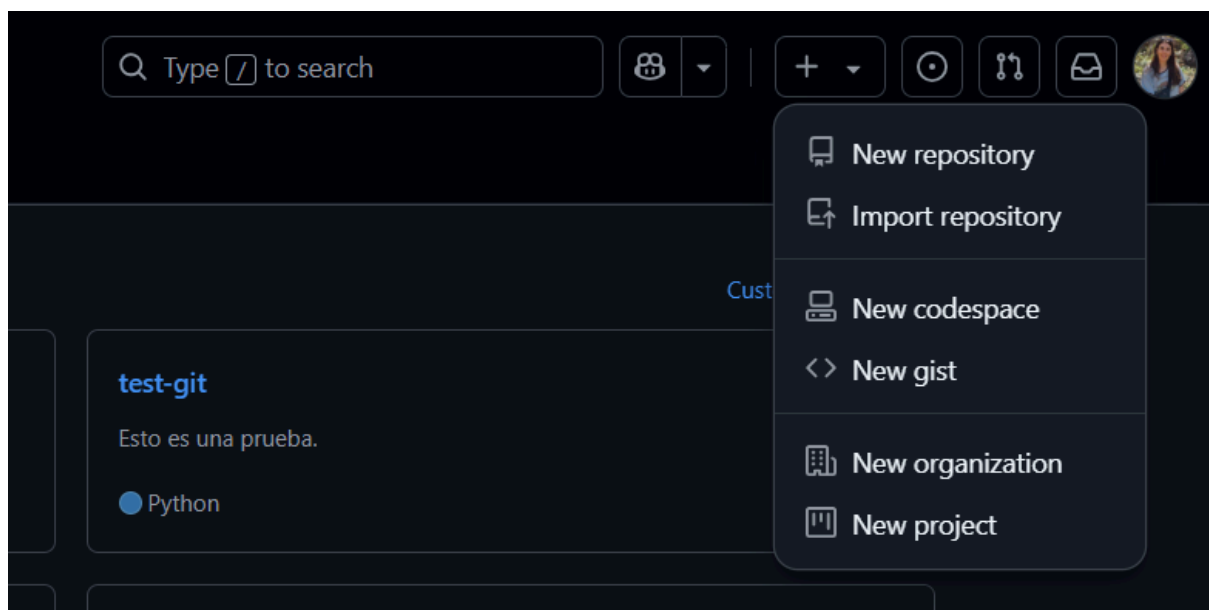
- **¿Qué es GitHub?**

Github es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador utilizando el sistema de control de versiones (Git, en este caso)

Es una herramienta de código abierto y gratuita en la cual no solo se aloja el código de diferentes proyectos sino que además, como usuario uno podrá ver los diferentes repositorios públicos, descargarlos o clonarlos y trabajar con ellos.

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

Para poder crear un repositorio, primero hay que tener una cuenta creada. En caso de tenerla o ya haberla creado, hay que ubicarse en mi perfil de Github (ejemplo: <https://github.com/maccantoni>), hacer click en el icon + ubicado en el header y seleccionar la opción **"New repository"**



En el siguiente paso se setean las primeras configuraciones del repositorio como el nombre, una descripción, si será un repositorio público o privado, si se agrega un readme, etc.

New repository


Q Type [7] to search

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 [crispy-octo-fortnight](#) ?

Description (optional)

☒  **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.


 /

✓ **prueba-tp** is available.

Great repository names are short and memorable. Need inspiration? How about [bug-free-meme](#) ?

Description (optional)

☒  **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** ▾

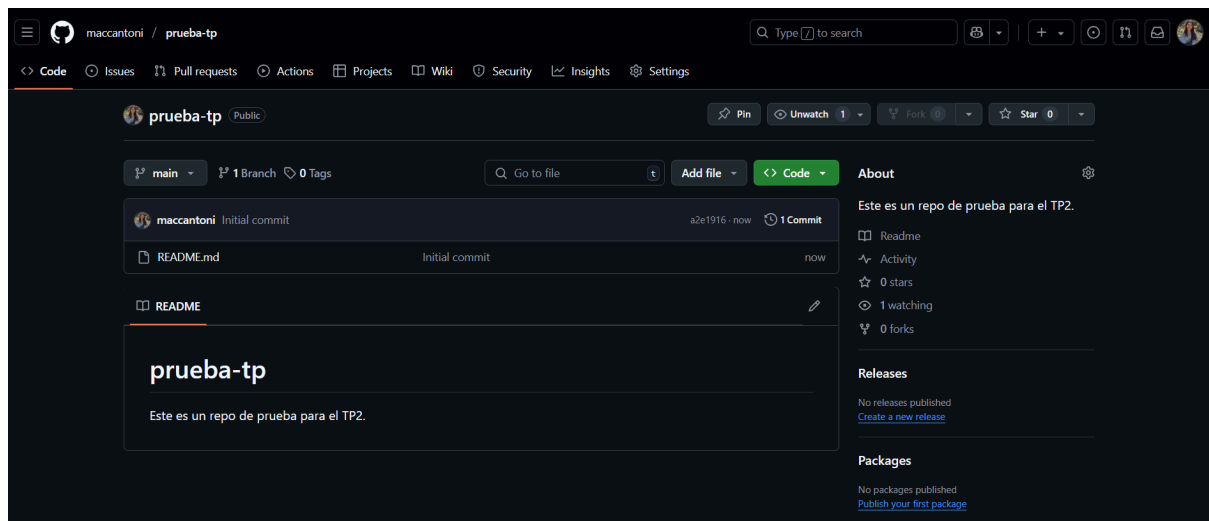
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

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

Clickear en “Create repository” y listo, el repositorio queda creado en Github.



- ¿Cómo crear una rama en Git?

Para crear un branch en Git, debemos partir de un archivo de código en el cual estemos trabajando. Por ejemplo:

Si estoy trabajando en el archivo ejercicio1.py y debo realizar y sumar algún cambio, puedo crear un nuevo branch para trabajar en forma más prolija y una vez que esté listo, unirlo o mergearlo al branch principal (main o master). Para esto se usa el comando **git branch nuevoBranch**

- ¿Cómo cambiar a una rama en Git?

Para cambiar de un branch a otro se usa el comando **git checkout nombreRama**. Para esto puede ser útil utilizar antes el comando **git branch**, para ver en qué branch estamos “parados” trabajando.

- ¿Cómo fusionar ramas en Git?

Para unir o fusionar branches, primero tenemos que ir al branch al cual le queremos fusionar contenido. Por ejemplo, si estamos trabajando en el branch ejercicio1, debemos ir al branch master o main con el comando **git checkout master**. Una vez allí, ejecutamos el comando **git merge ejercicio1**, es decir, mergeamos el contenido del nuevo branch en el branch original y/o principal.

- ¿Cómo crear un commit en Git?

Para crear un commit en Git, se utiliza el comando **git commit -m** donde -m es un mensaje que se le puede agregar entre comillas dando una breve explicación de lo que se está commiteando. Por ejemplo: **git commit -m “Se definen las variables”**. Este comando se usa para guardar cambios en el repositorio.

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

Luego de realizar los cambios que sean necesarios en el archivo que uno esté trabajando, se utiliza el comando **git add** . para sumar esos archivos al ambiente de stage, luego se hace un **git commit -m** para terminar de guardar los cambios.

- **¿Qué es un repositorio remoto?**

Un repositorio remoto es una versión de un proyecto que está hosteada en la web. Dependiendo del sitio donde se aloje, los repositorios pueden ser públicos o privados, y pueden descargarse o clonarse para trabajar con ellos en cualquier máquina.

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

Para agregar un repositorio remoto a Git hay que usar el comando **git remote add origin +** url del repositorio que se desea agregar.

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

Para empujar cambios a un repositorio remoto, se debe usar el comando **git push** para enviarlos a Github. Cuando se realiza esto por primera vez, se usa el comando **git push -u origin master**.

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

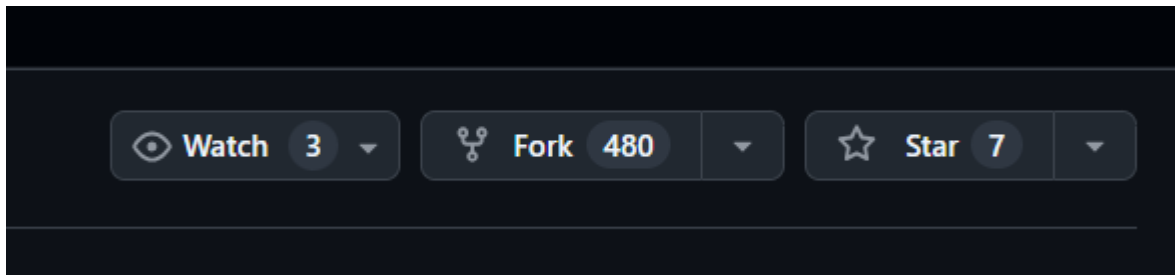
Para traer la versión más actualizada de un repositorio remoto a un proyecto local, hay que usar el comando **git pull origin master**. Es recomendable realizar esto, además, antes de pushear un cambio hacia un repositorio remoto, para asegurarse de tener y estar trabajando sobre la última versión.

- **¿Qué es un fork de repositorio?**

Un fork es un nuevo repositorio que comparte código y ajustes de visibilidad con el repositorio original. Suelen ser usados para realizar cambios o iterar sobre ideas que están en el repositorio original, del cual uno hace una copia y puede trabajar sobre el.

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

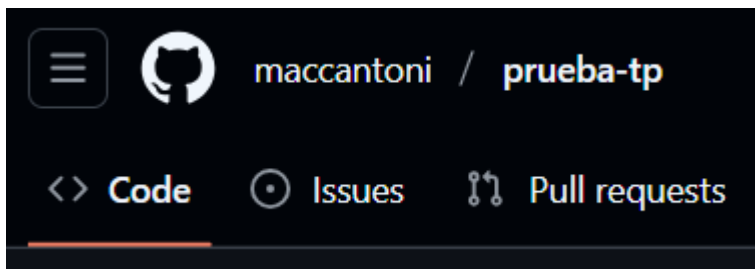
Para crear un fork de un repositorio, primero debemos ir al repositorio que nos interese. Una vez situado allí, clicar en el botón con la opción "Fork"



Luego, se completan algunas configuraciones como el nombre del fork y una breve descripción. Clickeamos en “Fork” y aguardamos a que el repositorio se cree en nuestra cuenta de Github.

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

Debemos ir a la pestaña “Pull requests” y crear uno nuevo. Allí veremos una pantalla donde se compara el trabajo y cambios que se realizaron en el branch que estoy trabajando comparado con el main o master. Clickeamos en “Create pull request” y allí tendremos que completar un breve comentario sobre el asunto y los cambios realizados.



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

El “dueño” del repositorio es quien recibirá el pull request. Este observará la solicitud con sus comentarios y evaluará si se aprueba y mergea o todavía no. En esta etapa también se pueden dejar comentarios en respuesta a los del pull request recibido. En caso que todo esté correcto, se aprobará el pull request y mergearan los cambios.

- **¿Qué es una etiqueta en Git?**

Las etiquetas en Git marcan un punto específico dentro de un proyecto como importantes. Se utilizan frecuentemente para marcar versiones de lanzamientos.

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

Para crear una etiqueta hay que tener en cuenta que hay dos opciones: ligeras y anotadas.

Las anotadas se guardan en la base de datos de Git como objetos enteros. Tienen un checksum; contienen el nombre del etiquetador, correo electrónico y fecha; tienen un mensaje asociado y pueden ser firmadas y verificadas. Para crear una de este tipo se usa el

comando **-a git tag**. Va acompañado de un mensaje indicado por el comando **-m** que especifica el mensaje de la etiqueta y también se guarda.

Para crear una etiqueta ligera solo se debe usar el comando **git tag**.

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

Se debe usar el comando **git push origin [etiqueta]**. En caso que se quieran enviar varios tags a la vez, se puede utilizar el comando **git push --tags**. Esto enviará todos los tags que todavía no existan en el repositorio remoto.

- **¿Qué es un historial de Git?**

El historial de Git muestra todos los cambios que se hayan realizado en el proyecto.

- **¿Cómo ver el historial de Git?**

Para ver el historial de Git se debe usar el comando **git log**, que mostrará los cambios desde los más recientes a los más antiguos.

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

Se utiliza el comando **git grep**, que permite buscar fácilmente a través de cualquier árbol o directorio de trabajo con commit por una cadena o expresión regular. A su vez, el comando **git log** se puede utilizar para encontrar commits específicos por el contenido de sus mensajes o incluso el contenido de las diferencias que introducen.

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

Para deshacer cambios se puede utilizar el comando **git reset**. Por ejemplo, si utilizamos **git reset [archivo]** se quitarán los archivos de stage.

Otros comandos reset:

git reset --modo HEAD^ → Volver a commit anterior

git reset --modo HEAD^N → Volver hacia el N° anterior commit

git reset --modo hash-commit → Volver hacia commit específico

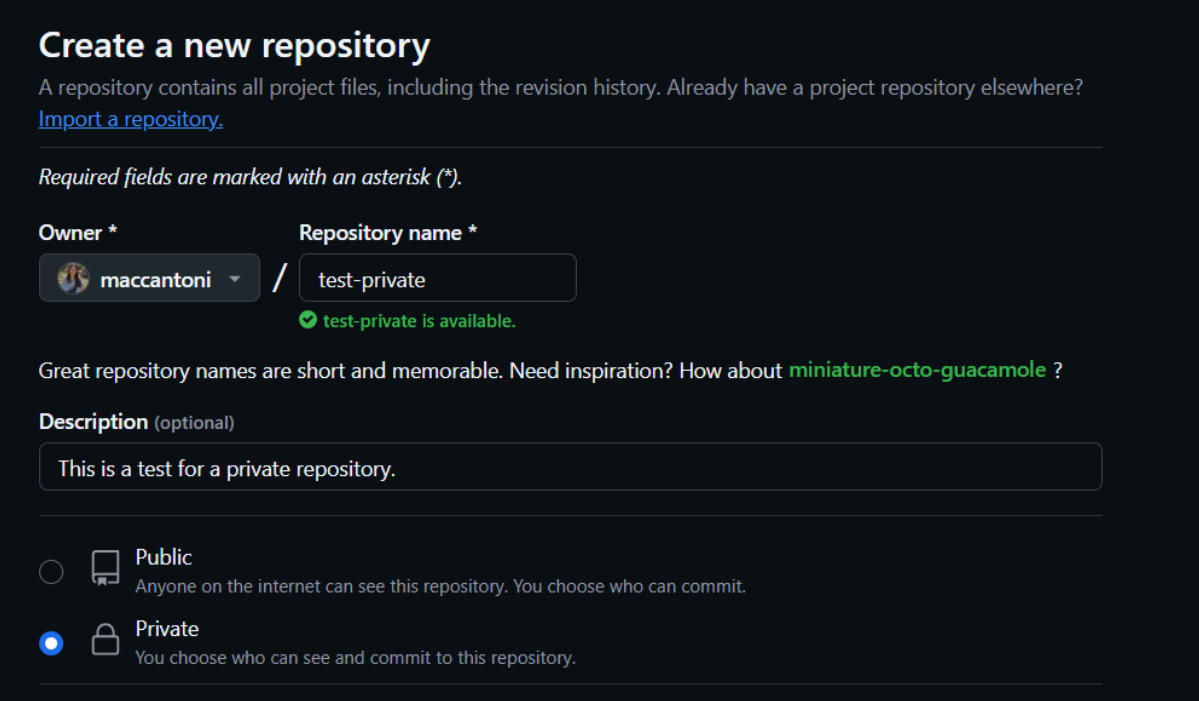
- **¿Qué es un repositorio privado en GitHub?**

Un repositorio privado es aquel que tiene restricciones en cuanto a quién puede ver y hacer commits sobre él. Quien definirá esto es el dueño del repositorio seteando esta opción en la configuración al crear un nuevo repositorio en Github.

• ¿Cómo crear un repositorio privado en GitHub?

Para crear un nuevo repositorio privado en Github, primero que nada se debe estar logueado con una cuenta en el sitio. Luego, clickear en el icon + y seleccionar la opción "New repository" (<https://github.com/new>)

En el siguiente paso se encontrarán las configuraciones iniciales, dentro de las cuales está la privacidad. Setear la opción "Private" y clickear 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 * **Repository name ***


 maccantoni / test-private


✔ test-private is available.

Great repository names are short and memorable. Need inspiration? How about [miniature-octo-guacamole](#) ?

Description (optional)

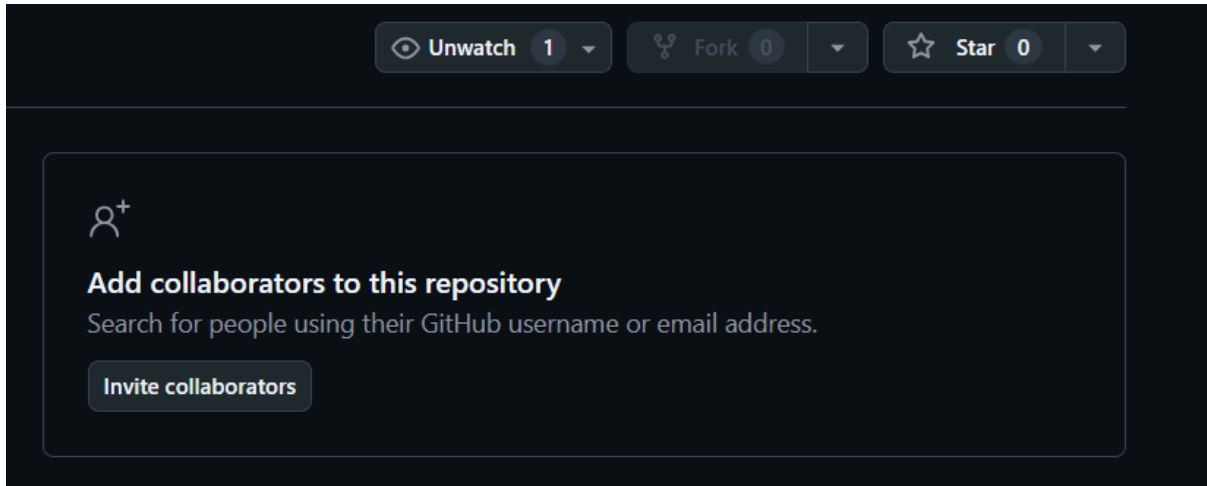
This is a test for a private repository.

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

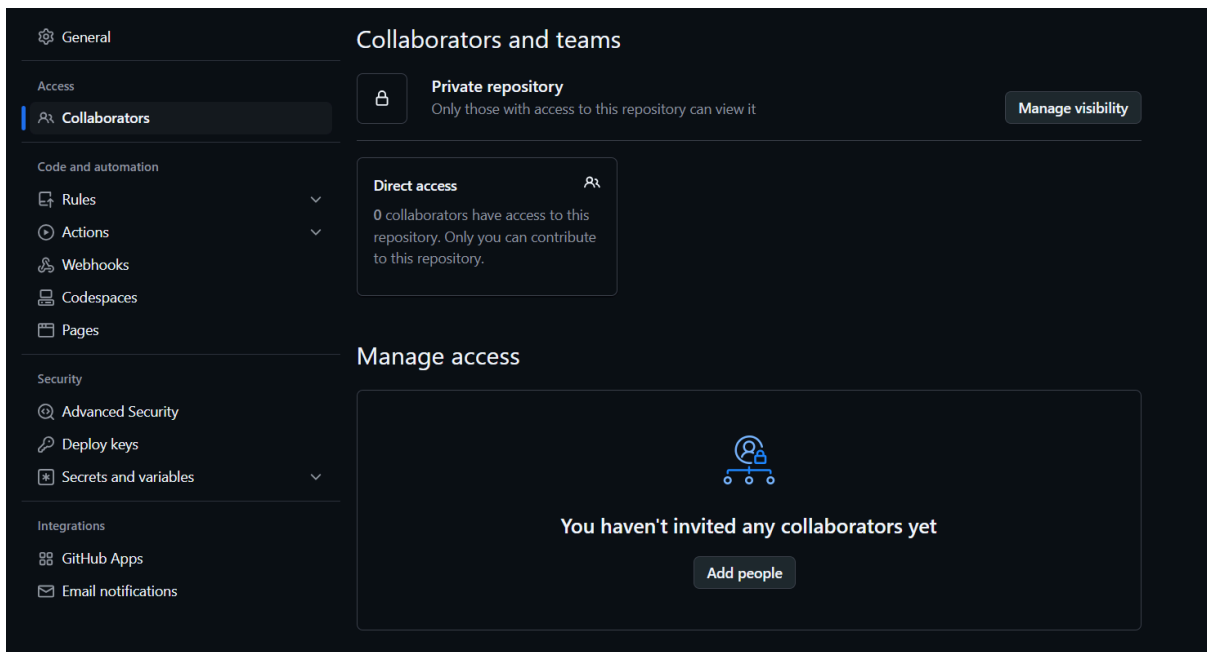
☒  **Private**
You choose who can see and commit to this repository.

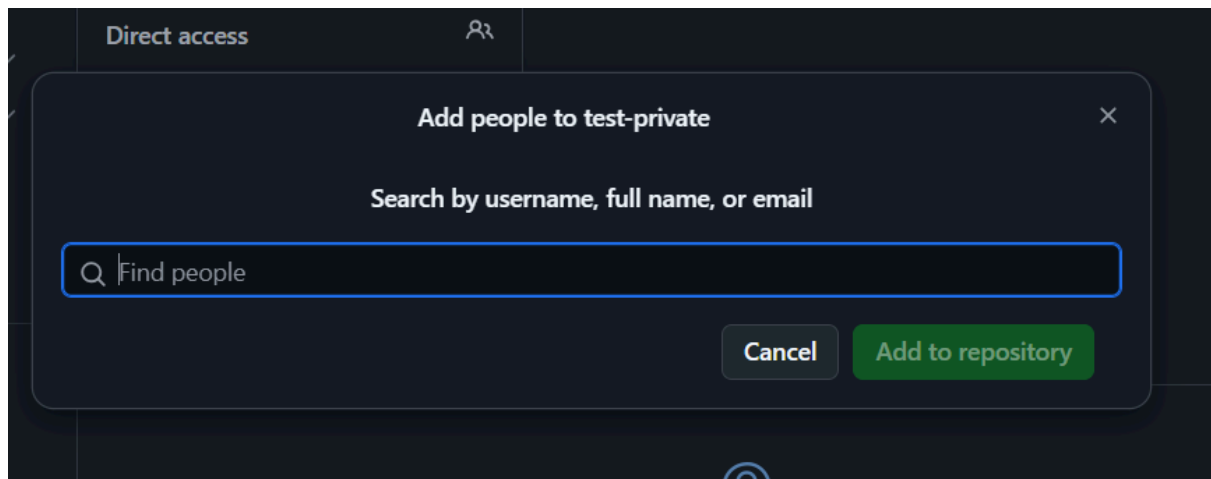
• ¿Cómo invitar a alguien a un repositorio privado en GitHub?

En caso de querer sumar colaboradores, se puede hacerlo desde la configuración del proyecto. Esto hará que los colaboradores tengan permisos para hacer “push”, por lo tanto, tendrán tanto acceso de lectura como de escritura en el proyecto y en el repositorio Git.



Si clickeamos en “Invite collaborators” y luego en “Add people” se podrá agregar a quien uno quiera darle permisos buscando por mail, nombre o username en Github. Además se deberá seleccionar que tipo de acceso se le otorgará a ese usuario.





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

Un repositorio público en Github es un repositorio que todo el mundo que ingrese al link puede ver y descargar, clonar o hacer fork. Uno como dueño es quien define o acepta cambios que cualquier otra persona pueda hacer.

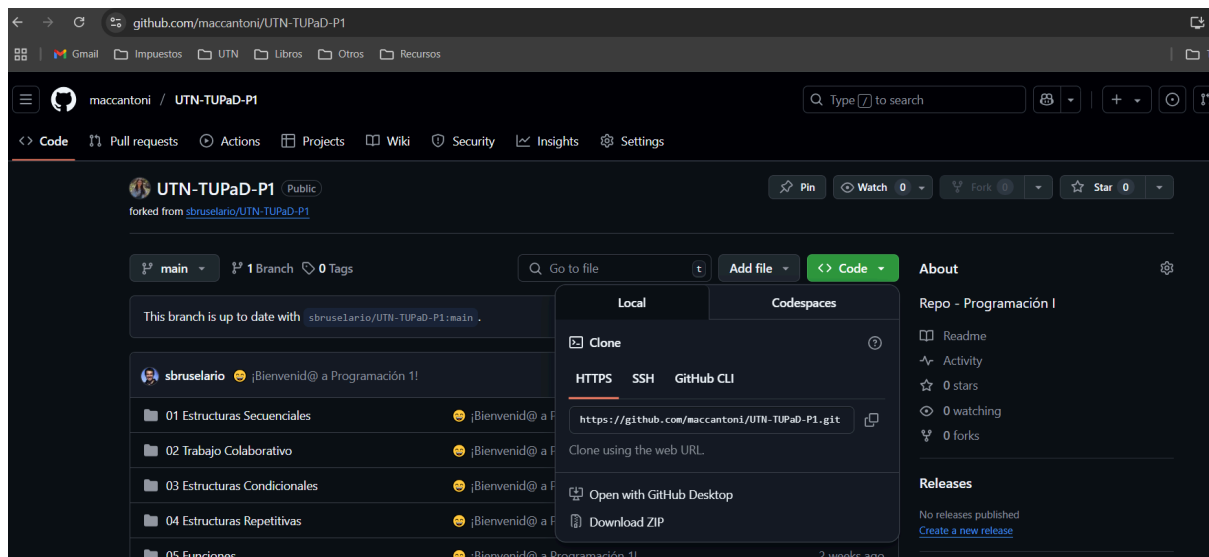
- **¿Cómo crear un repositorio público en GitHub?**

Para crear un repositorio público, se deben seguir los mismos pasos que en el punto anterior:

1. Iniciar sesión en Github
2. Clickear en el icon + (New repository)
3. Seleccionar las configuraciones iniciales, incluyendo la de privacidad, que debe quedar como "público"
4. Clickear en "Create repository"

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

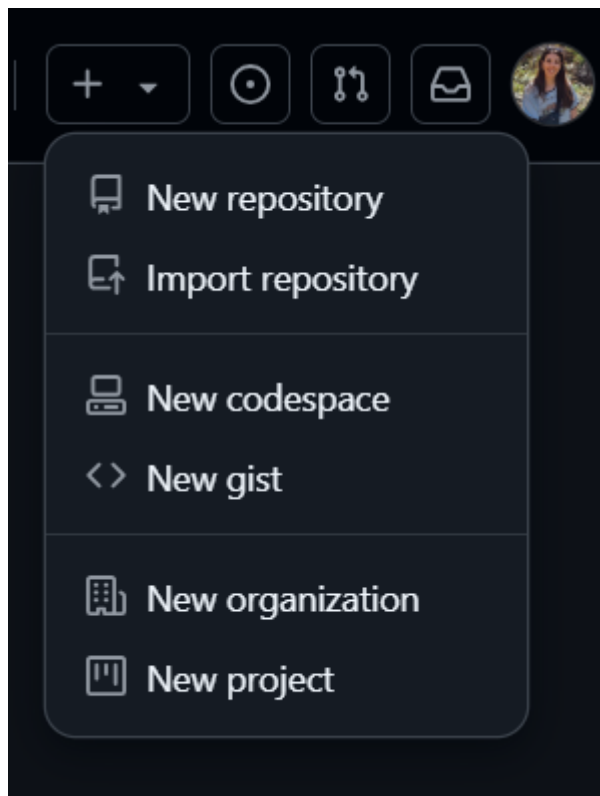
Para compartir un repositorio público solo es necesario el link del mismo. Se puede copiar directamente la URL mientras estamos situados sobre el proyecto, o clickear en "code" y en el menú que se despliega, copiar la URL HTTPS.



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 *  maccantoni / Repository name * RepoTP2
 RepoTP2 is available.

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

Description (optional)

Este es un repo para el TP2.

- ☒  **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.](#)

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).

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2> git add
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .' ?
hint: Disable this message with 'git config set advice.addEmptyPathsSpec false'
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2> git add .
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2> git commit -m "Este es un archivo de texto para el TP 2"
[main a85845c] Este es un archivo de texto para el TP 2
 1 file changed, 1 insertion(+)
 create mode 100644 mi-archivo.txt
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2>
```

This PC > Desktop > UTN > TUP > Programación I > Ejercicios > TP 2 > RepoTP2						Search RepoTP2
Name	Status	Date modified	Type	Size		
 mi-archivo		1/4/2025 17:17	Text Document	1 KB		
 README		1/4/2025 17:15	Markdown Source ...	1 KB		

RepoTP2Public

PinUnwatch1

main1 Branch0 Tags

Go to file

Add fileCode

maccantoniInitial commitc9ed2af · 6 minutes ago1 Commit

README.mdInitial commit6 minutes ago

README

RepoTP2

Este es un repo para el TP2.

```
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2> git branch
* main
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2> git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 382 bytes | 95.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/maccantoni/RepoTP2
c9ed2af..a85845c main -> main
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2>
```

RepoTP2Public

PinUnwatch1Fork0Star0

main1 Branch0 Tags

Go to file

Add fileCode

maccantoniEste es un archivo de texto para el TP 2a85845c · 3 minutes ago2 Commits

README.mdInitial commit8 minutes ago

mi-archivo.txtEste es un archivo de texto para el TP 23 minutes ago

README

RepoTP2

Este es un repo para el TP2.

About

Este es un repo para el TP2.

ReadmeActivity0 stars1 watching0 forks

ReleasesNo releases publishedCreate a new release

PackagesNo packages publishedPublish your first package

Creando Branchs

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

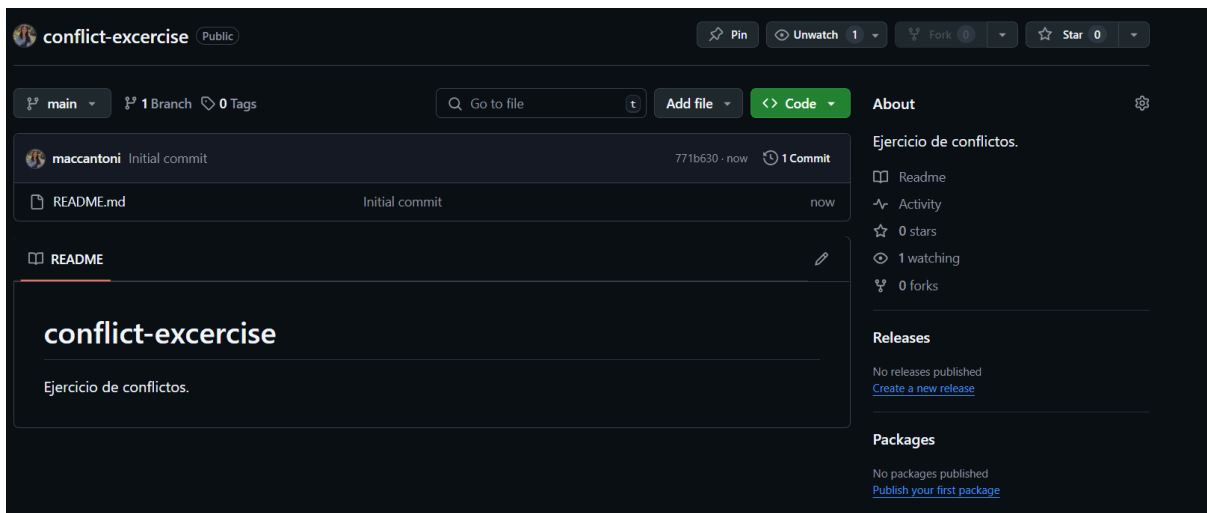
```
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2> git checkout -b nuevaRama
Switched to a new branch 'nuevaRama'
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2>
```

```
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2> git add .
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2> git commit -m "Modifico el archivo en una nueva branch"
[nuevaRama 4206fdb] Modifico el archivo en una nueva branch
1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2>
```

```
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2> git status
On branch nuevaRama
nothing to commit, working tree clean
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2> git push origin nuevaRama
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 404 bytes | 101.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'nuevaRama' on GitHub by visiting:
remote:   https://github.com/maccantoni/RepoTP2/pull/new/nuevaRama
remote:
To https://github.com/maccantoni/RepoTP2
 * [new branch]   nuevaRama -> nuevaRama
PS C:\Users\macac\OneDrive\Escritorio\UTN\TUP\Programación I\Ejercicios\TP 2\RepoTP2>
```

3) Realizar la siguiente actividad:

1. Crear un repositorio en GitHub



2. Clonar el repositorio a tu máquina local

```
macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2 (master)
$ git clone https://github.com/maccantoni/conflict-exercise.git
Cloning into 'conflict-exercise'...
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.

macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2 (master)
$ |
```

3. Crear una nueva rama y editar un archivo.

```
macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (feature-branch)
$ |
```

```
❖ README.md > # conflict-exercise
1  # conflict-exercise
2  Ejercicio de conflictos.
3  | Esto es un cambio en la branch feature-branch.
4
```

```
macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (feature-branch)
$ git branch
* feature-branch
  main

macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (feature-branch)
$ git add README.md

macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (feature-branch)
$ git commit -m "Added a new line in feature-branch"
[feature-branch 206ffed] Added a new line in feature-branch
1 file changed, 1 insertion(+)
```

4. Volver a la rama principal y editar el mismo archivo

```
macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (main)
$ |
```

```
1 README.md > # conflict-exercise
2 # conflict-exercise
3 Ejercicio de conflictos.
4 Este es otro cambio pero en la branch main.
```

5. Hacer un merge y generar un conflicto

```
macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (main)
$ git add README.md

macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main e686c60] Added a line in main branch
1 file changed, 1 insertion(+)

macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (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.

macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (main|MERGING)
$ |
```

```
1 README.md > # Ejercicio de conflictos.<<<<<< HEAD
2 # Ejercicio de conflictos.<<<<<< HEAD
3 Ejercicio de conflictos.<<<<<< HEAD
4 Este es otro cambio pero en la branch main.
5 =====
6 Esto es un cambio en la branch feature-branch.
7 >>>>>> feature-branch (Incoming Change)
8
```

6. Resolver el conflicto

```
❗ README.md > [abc] # conflict-exercise
1  # conflict-exercise
2  Ejercicio de conflictos.
3  Esto es un cambio en la branch feature-branch.
4
5
```

7. Subir los cambios a GitHub

```
macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (main|MERGING)
$ git add README.md

macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main 5bc1433] Resolved merge conflict

macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 818 bytes | 204.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/maccantoni/conflict-exercise.git
   771b630..5bc1433  main -> main

macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (main)
$ git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/maccantoni/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/maccantoni/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch

macac@DESKTOP-VRAFA03 MINGW64 ~/OneDrive/Escritorio/UTN/TUP/Programación I/Ejercicios/TP 2/conflict-exercise (main)
$ |
```


8. Verificar en GitHub

The screenshot shows the GitHub repository 'conflict-exercise' (Public). The interface is in dark mode. At the top, there are buttons for 'Pin', 'Unwatch' (1), 'Fork' (0), and 'Star' (0). Below the repository name, there are tabs for 'main' (selected), '2 Branches', and '0 Tags'. A search bar 'Go to file' and buttons 'Add file' and '<> Code' are visible. A 'Switch branches/tags' modal is open on the left, showing a search bar and a list of branches: 'main' (selected, default) and 'feature-branch'. The main content area shows a 'Resolved merge conflict' message with a commit hash '5bc1433' and a timestamp '1 minute ago'. Below this, there is a text box containing the text 'Ejercicio de conflictos. Esto es un cambio en la branch feature-branch.' The right sidebar contains the 'About' section with the title 'Ejercicio de conflictos.', a 'Readme' icon, and links for 'Activity', '0 stars', '1 watching', and '0 forks'. Below this are sections for 'Releases' (No releases published, Create a new release) and 'Packages' (No packages published, Publish your first package).

The screenshot shows the same GitHub repository 'conflict-exercise' (Public). The interface is in dark mode. At the top, there are buttons for 'Pin', 'Unwatch' (1), 'Fork' (0), and 'Star' (0). Below the repository name, there are tabs for 'main' (selected), '2 Branches', and '0 Tags'. A search bar 'Go to file' and buttons 'Add file' and '<> Code' are visible. The main content area shows a 'Resolved merge conflict' message with a commit hash '5bc1433' and a timestamp '2 minutes ago'. Below this, there is a list of files: 'README.md' (Resolved merge conflict, 2 minutes ago). The 'README' file is selected, and its content is displayed in a text box: 'conflict-exercise' followed by 'Ejercicio de conflictos. Esto es un cambio en la branch feature-branch.' The right sidebar contains the 'About' section with the title 'Ejercicio de conflictos.', a 'Readme' icon, and links for 'Activity', '0 stars', '1 watching', and '0 forks'. Below this are sections for 'Releases' (No releases published, Create a new release) and 'Packages' (No packages published, Publish your first package).