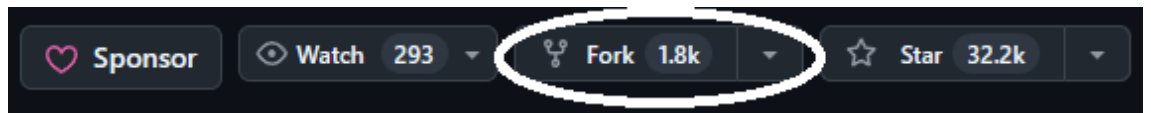


1)

- GitHub es una plataforma basada en la nube donde puedes almacenar, compartir y trabajar junto con otros usuarios para escribir código. Además, GitHub trabaja con Git, que es un sistema de control de versiones que realiza un seguimiento de los cambios en los archivos.
- Para poder crear un nuevo repositorio en GitHub, primero se debe contar con una cuenta en la plataforma. Luego, en la pantalla principal, se debe apretar en el signo + que se encuentra en la parte superior, a la derecha. Una vez se abra el menú contextual, se deberá hacer click en el botón “*New repository*”. Cuando hagamos click aquí, se nos abrirá una página que nos permitirá darle un nombre a nuestro nuevo repositorio, agregar una descripción, configurar la visibilidad del repositorio, entre otras configuraciones. Por último, y una vez se haya creado el repositorio, GitHub nos proporcionará los comandos necesarios para sincronizar nuestro repositorio remoto con el local.
- Para crear una rama, deberemos ubicarnos en nuestro repositorio local y abrir una consola en esa ubicación. Una vez se abra la consola, ejecutaremos el comando “`git branch nombre-rama`”.
- Para cambiar de rama, abriremos la consola desde la carpeta de nuestro repositorio local y luego ejecutaremos el comando “`git checkout nombre-rama`”. De esta manera, estaremos ubicados en la rama seleccionada.
- Para fusionar ramas primero debemos posicionarnos en la rama de destino, con el comando “`git checkout rama-destino`” para luego ejecutar “`git merge rama-origen`”. De esta manera, obtendremos los cambios de la rama-origen en la rama-destino.
- Para crear un commit, primero se deberán agregar los cambios con el comando “`git add .`” si deseamos agregar todos los archivos modificados o “`git add archivo1 archivo2 archivo3...`” si lo que deseamos es agregar solo determinados archivos. Una vez realizado esto, podremos ejecutar el comando “`git commit -m “descripcion del commit”`” para poder guardar los cambios realizados.
- Para enviar un commit, primero lo debemos tener ya realizado, como se explicó en el punto anterior y luego ejecutar “`git push`” situados en la rama que queremos enviar hacia el repositorio remoto. Esto funcionará solo si previamente se realizó la configuración necesaria para sincronizar nuestro repositorio local con el remoto. Además, la primera vez es recomendable utilizar el comando “`git push -u origin main`” para que git entienda que siempre que enviemos un cambio desde la rama main, irá a la rama origin del repositorio remoto y luego solo utilizar el comando que se explicó primeramente.

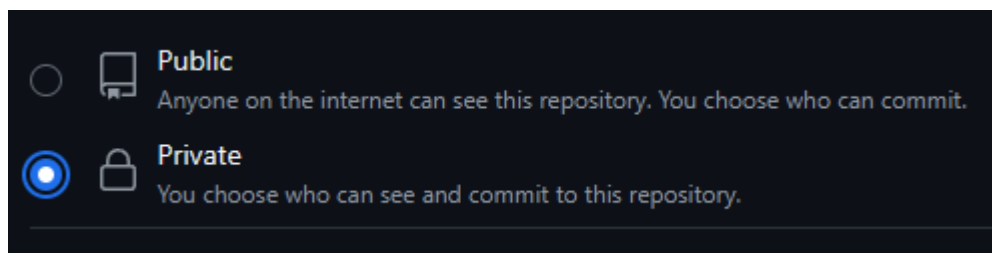
- Un repositorio remoto es aquel que está alojado en la nube, es decir, en cualquier plataforma de internet que pueda almacenar repositorios en la red, en el caso de este curso, puede ser GitHub.
- Para agregar un repositorio remoto a Git se deben seguir las instrucciones que muestra GitHub al momento de crear el repositorio remoto. Estas son, intuyendo que ya se inicializó el repositorio:
 - `git commit -m "first commit"`
 - `git branch -M main`
 - `git remote add origin`
`https://github.com/IgnacioVillordo/prueba.git`
 - `git push -u origin main`
- Para empujar cambios al repositorio remoto se utiliza el comando “git push”.
- Para tirar de cambios de un repositorio remoto, se utiliza el comando “git pull”.
- Un fork es una copia de un repositorio en la cuenta de un usuario, de esta manera puede hacer los cambios que crea necesario, sin modificar el repositorio original.
- Para crear un fork de un repositorio en GitHub, es tan simple como ir al repositorio al que se le quiere realizar el fork y apretar el botón que dice fork, que se encuentra en la parte superior de la página.



- Para crear una solicitud de extracción primero se necesita cumplir con ciertos requisitos, entre ellos tenemos la existencia de un fork del repositorio con el que se quiere colaborar y los cambios que se quieren realizar guardados en un commit. Además, siempre es bueno leer el archivo contributing.md o la parte de contributing del archivo readme.md, estos contienen pautas para la colaboración como así también limitaciones y recomendaciones. Una vez tengamos todo esto, ejecutaremos “git push” para subir nuestro commit al fork. Con el commit ya en el fork remoto, si entramos a la página de github, nos aparecerá un cartel que dice “Compare & pull request”, si hacemos click aquí, nos llevará a un apartado para realizar un comentario (opcional) sobre los cambios que hemos realizado. Una vez enviado el pull request, será trabajo del autor original aceptar o no nuestros cambios.
- Para aceptar una solicitud de extracción, debemos ir a nuestro repositorio y en las pestañas superiores, debajo del nombre de nuestro repositorio, apretar en donde dice pull requests. Se nos abrirá una

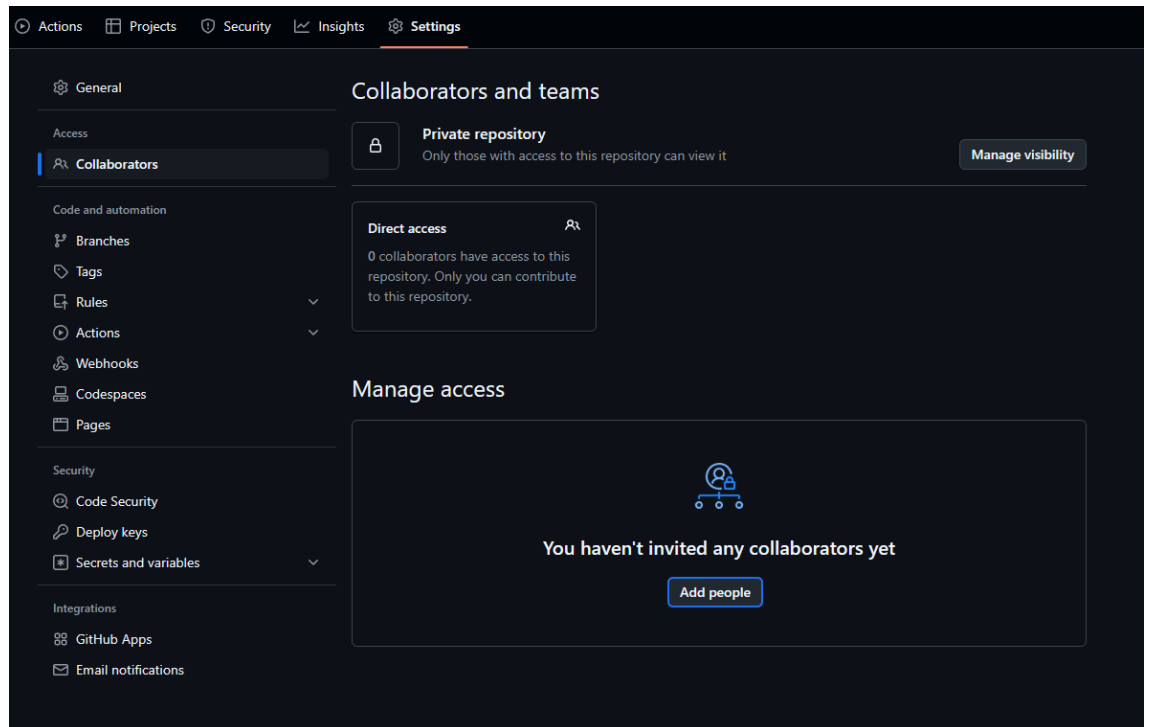
página que nos permitirá ver los cambios realizados en nuestro repositorio por el creador de la PR. Ahí tendremos la posibilidad de realizar un comentario al desarrollador y aceptar o rechazar el PR.

- Una etiqueta marca un punto específico en el historial de un repositorio.
- Para crear una etiqueta ligera en Git, debemos ejecutar el comando “git tag nombre-etiqueta”. Para crear una etiqueta anotada, debemos utilizar “git tag -a nombre-etiqueta “descripcion””. Esta última etiqueta nos permite también apuntar la etiqueta a un commit en específico, simplemente agregando al final el hash del commit deseado.
- Para enviar una etiqueta hacia GitHub debemos ejecutar “git push origin nombre-etiqueta”.
- Un historial de Git es un registro cronológico que incluye todos los commits que hemos hecho, desde su creación, hasta la actualidad.
- Para poder ver el historial debemos ejecutar el comando “git log”.
- Para borrar el historial de Git se utiliza el comando “git reset”. Este comando tiene tres modos distintos:
 - --soft: mueve el puntero al commit especificado, pero los archivos no se modifican ni el staging, que se genera cuando uno ejecuta git add.
 - --mixed: mueve el puntero y elimina el staging, pero no modifica los archivos.
 - --hard: mueve el puntero, elimina el staging y modifica los archivos.
- Un repositorio privado es aquel al que solo pueden acceder el propietario y las personas a las que este de acceso.
- Para crearlo es tan simple como seleccionar la opción private, como se ve a continuación:

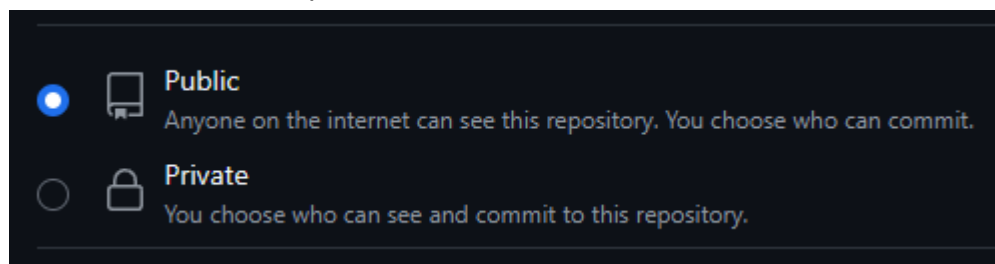


- Para permitir el acceso a nuestro repositorio a alguien, debemos ir a la pestaña de settings y luego a la sección de collaborators. Una vez allí, nos aparecerá un botón “Add people” que deberemos apretar para que nos aparezca una ventana en la que podremos buscar a la persona que

se quiere agregar al proyecto.



- Un repositorio público es aquel que es visible para cualquier persona.
- Para crear un repositorio público, debemos seleccionar la opción “Public” al crear un repositorio, como se muestra a continuación:



Cabe aclarar que, al crear un repositorio en GitHub, esta es la opción por defecto.

- Para compartir nuestro repositorio solamente debemos compartir el link de este.

2)

- Git creado con el nombre TP2 y se inicializa con el archivo readme.md.


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 *

 IgnacioVillordo

 /


TP2

 TP2 is available.


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

Description (optional)

Repositorio Trabajo Práctico 2 - 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.

- Archivo creado y commit subido a GitHub.

```
C:\Users\ignac\Documents\Programacion 1>git clone https://github.com/IgnacioVillordo/TP2.git
Cloning into 'TP2'...
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.

C:\Users\ignac\Documents\Programacion 1>cd TP2

C:\Users\ignac\Documents\Programacion 1\TP2>echo "Texto de ejemplo - Trabajo Practico 2" > mi-archivo.txt

C:\Users\ignac\Documents\Programacion 1\TP2>git add .

C:\Users\ignac\Documents\Programacion 1\TP2>git commit -m "Agregando mi-archivo.txt"
[main b0e2773] Agregando mi-archivo.txt
 1 file changed, 1 insertion(+)
 create mode 100644 mi-archivo.txt

C:\Users\ignac\Documents\Programacion 1\TP2>git push origin main
info: please complete authentication in your browser...
git: 'credential-manager-core' is not a git command. See 'git --help'.
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), 342 bytes | 342.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/IgnacioVillordo/TP2.git
   ba7435d..b0e2773  main -> main

C:\Users\ignac\Documents\Programacion 1\TP2>
```

main 1 Branch 0 Tags

Go to file Add file Code

IgnacioVillordo Agregando mi-archivo.txt b0e2773 · 5 minutes ago 2 Commits

README.md	Initial commit	18 minutes ago
mi-archivo.txt	Agregando mi-archivo.txt	5 minutes ago

TP2 / mi-archivo.txt

IgnacioVillordo Agregando mi-archivo.txt

Code Blame 1 lines (1 loc) · 41 Bytes

```
1 "Texto de ejemplo - Trabajo Practico 2"
```

- Se creó un segundo archivo y luego fue enviado a GitHub

```
C:\Users\ignac\Documents\Programacion 1\TP2>git checkout -b rama-2
Switched to a new branch 'rama-2'

C:\Users\ignac\Documents\Programacion 1\TP2>echo "Segundo archivo creado" > archivo2.txt

C:\Users\ignac\Documents\Programacion 1\TP2>git add .

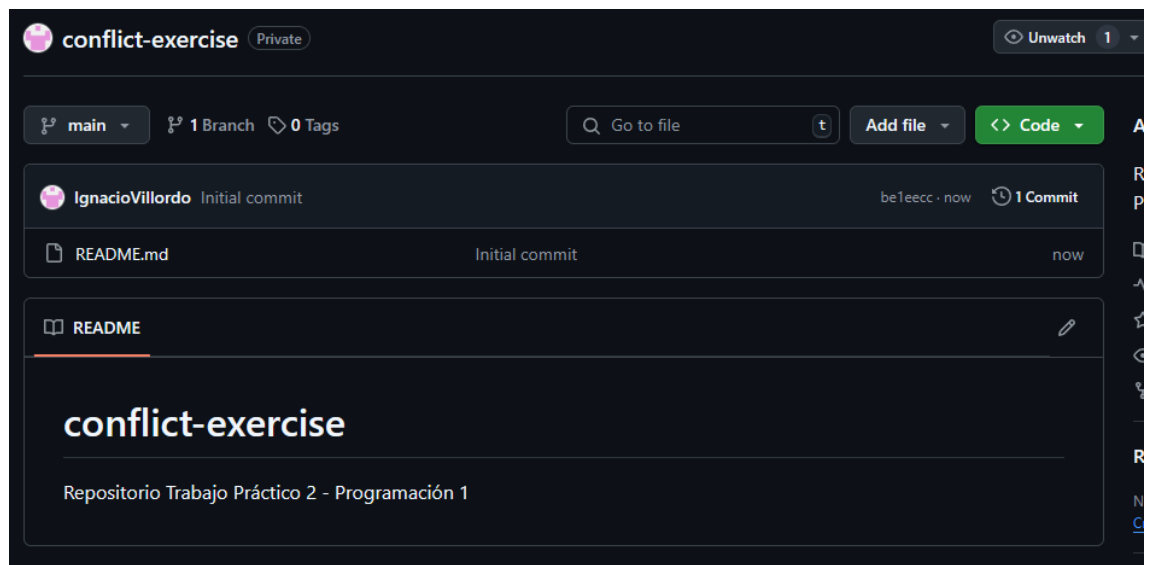
C:\Users\ignac\Documents\Programacion 1\TP2>git commit -m "Se agrega archivo2.txt"
[rama-2 c6492dd] Se agrega archivo2.txt
1 file changed, 1 insertion(+)
create mode 100644 archivo2.txt

C:\Users\ignac\Documents\Programacion 1\TP2>git push origin rama-2
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), 355 bytes | 355.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'rama-2' on GitHub by visiting:
remote:   https://github.com/IgnacioVillordo/TP2/pull/new/rama-2
remote:
To https://github.com/IgnacioVillordo/TP2.git
 * [new branch]      rama-2 -> rama-2
```

Link al repositorio <https://github.com/IgnacioVillordo/TP2-Actividad2>

3)

- Paso 1: repositorio creado con el archivo readme.md correspondiente.



- Paso 2: repositorio clonado con la consola ubicada en la carpeta del repositorio.

```
C:\Users\ignac\Documents\Programacion 1>git clone https://github.com/IgnacioVillordo/conflict-exercise
Cloning into 'conflict-exercise'...
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.

C:\Users\ignac\Documents\Programacion 1>cd conflict-exercise
```

- Paso 3: nueva rama creada.

```
C:\Users\ignac\Documents\Programacion 1\conflict-exercise>git checkout -b feature-branch
Switched to a new branch 'feature-branch'

C:\Users\ignac\Documents\Programacion 1\conflict-exercise>git add README.md

C:\Users\ignac\Documents\Programacion 1\conflict-exercise>git commit -m "Segunda linea agregada en feature-branch"
[feature-branch a7b4867] Segunda linea agregada en feature-branch
1 file changed, 1 insertion(+)

C:\Users\ignac\Documents\Programacion 1\conflict-exercise>
```

Archivo editado.

```

i README.md X
C: > Users > ignac > Documents > Programacion 1 > conflict-exercise > i README.md
1 # conflict-exercise
2 Repositorio Trabajo Práctico 2 - Programación 1
3 Segunda linea agregada.

```

- Paso 4: se cambia de rama, se edita el archivo y se crea un nuevo commit, ahora desde la rama main.

```

C: > Users > ignac > Documents > Programacion 1 > conflict-exercise > i README.md
1 # conflict-exercise
2 Repositorio Trabajo Práctico 2 - Programación 1
3 Línea agregada en rama main.

```

```
C:\Users\ignac\Documents\Programacion 1\conflict-exercise>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\ignac\Documents\Programacion 1\conflict-exercise>git add README.md

C:\Users\ignac\Documents\Programacion 1\conflict-exercise>git commit -m "Línea agregada en rama main"
[main 2a9d891] Línea agregada en rama main
1 file changed, 1 insertion(+)

```

- Paso 5: se hace el merge.

```
C:\Users\ignac\Documents\Programacion 1\conflict-exercise>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: se resuelve el conflicto.

```
C:\Users\ignac\Documents\Programacion 1\conflict-exercise>git add README.md

C:\Users\ignac\Documents\Programacion 1\conflict-exercise>git commit -m "Se resuelve conflicto"
[main 4edc7cb] Se resuelve conflicto

```

```

1 v # conflict-exercise
2 Repositorio Trabajo Práctico 2 - Programación 1
3 Segunda linea agregada.

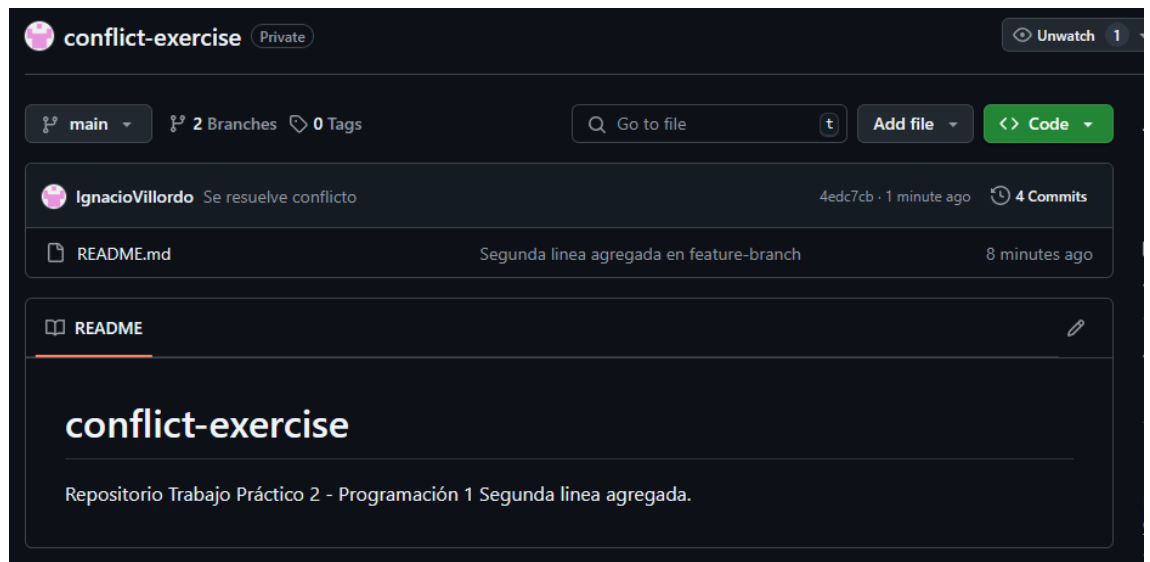
```


- Paso 7: se suben los cambios a ambas ramas.

```
C:\Users\ignac\Documents\Programacion 1\conflict-exercise>git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 754 bytes | 754.00 KiB/s, done.
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/IgnacioVillordo/conflict-exercise
    beleecc..4edc7cb  main -> main

C:\Users\ignac\Documents\Programacion 1\conflict-exercise>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/IgnacioVillordo/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/IgnacioVillordo/conflict-exercise
 * [new branch]      feature-branch -> feature-branch
```

- Paso 8: se ve el texto “Segunda línea agregada” en el archivo README.md. Además, se ve la existencia de 2 ramas.



Link al repositorio: <https://github.com/IgnacioVillordo/TP2-Actividad3>.