

TP 2 - GIT/GITHUB

Link del repositorio

<https://github.com/k4ry32/UTN-TUPaD-P1.git>

Actividades

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

- **¿Qué es GitHub?**

Github es una plataforma basada en la nube donde se puede almacenar, compartir y trabajar junto a otros usuarios en proyectos de código. El principal uso es llevar un control de versiones (cambios) del código utilizando Git.

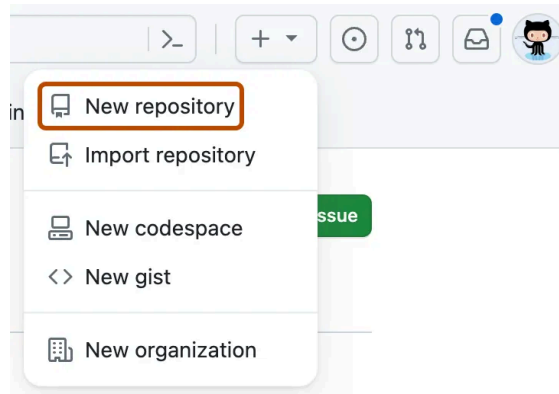
Está basado en repositorios, donde se sube por lo general un proyecto distinto por cada repositorio. Los principales beneficios están en que se puede tener un control preciso de quién puede tener acceso al código fuente, además de que facilita procesos de integración y deploy.

También existen repos donde tienen dentro más de un proyecto (mono-repositorio), esto facilita la documentación y la ubicación de todas las partes de un proyecto más grande: apis, front, etc. Entre los principales inconvenientes está que el acceso es total. Por lo que tener una asignación de permisos granular al código es más complicado.

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

Para crear un nuevo repositorio en nuestro perfil de Github, desde la web, es necesario seguir estos pasos:

1. En la esquina superior derecha de cualquier página, selecciona y luego haz clic en Nuevo repositorio.



2. Escriba un nombre corto y fácil de recordar para el repositorio. Por ejemplo: "hola-mundo".

Owner * Repository name *

octocat / hello-world ✓

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

Description (optional)

3. Opcionalmente, puede agregar una descripción del repositorio. Por ejemplo, "Mi primer repositorio en GitHub".
4. Elige la visibilidad del repositorio (Público o Privado).
5. Seleccione Initialize this repository with a README (Inicializar este repositorio con un archivo Léame). Este archivo es útil para hacer una descripción del proyecto, agregar instrucciones de instalación e inicialización o instrucciones de uso, etc.
6. Haga clic en Create repository (Crear repositorio).

Con estos pasos se habrá creado un nuevo repo con un archivo README dentro.

- **¿Cómo crear una rama en Git?**

Cada repo se inicia con una rama principal denominada "main" que es la que contiene el código ya testeado y funcionando correctamente.

Al momento de empezar una nueva tarea es recomendable crear una rama aparte (copia de main) donde se puede trabajar sin cambiar el código que ya está en producción, una vez testeado y completo se une la rama creada a la rama main.

Para crear una nueva rama se utiliza el comando de Git:

```
git branch <nombre_de_la_rama>
```

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

Luego de crear una nueva rama con el comando antes visto hay que posicionarse en ésta. Para eso se utiliza el comando:

```
git checkout <nombre_de_la_rama>
```

Una buena opción es usar el siguiente comando que crea y a su vez se cambia a la rama nueva:

```
git checkout -b <nombre_de_la_rama>
```

- **¿Cómo fusionar ramas en Git?**

Primero se debe posicionar en la rama a la cual se quiere fusionar otra rama, luego se utiliza el comando:

```
git merge <nombre_de_la_rama_a_fusionar>
```

Esto traerá a la rama actual todo el código nuevo y diferente que esté en la rama elegida.

- **¿Cómo crear un commit en Git?**

Luego de introducir nuevos cambios en uno o varios archivos se deben aceptar o almacenar los cambios primero con el comando:

```
git add <nombre_o_nombres_de_archivos_modificados>
```

También se pueden agregar todos los cambios de todo los archivos modificados con el comando:

```
git add .
```

Una vez hecho el paso anterior está listo para hacer el commit con el comando:

```
git commit -m "mensaje descriptivo de la modificación"
```

Se puede hacer los dos pasos en uno con el comando:

```
git commit -am "mensaje descriptivo de la modificación"
```

Esto almacena todos los cambios y crea el commit.

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

Luego de almacenar los cambios y crear el commit este se debe sincronizar con el repositorio en la nube (github), o sea, subir los cambios para que los demás miembros del equipo puedan verlos.

Para esto, si es la primera vez que subimos cambios, se debe ejecutar el comando:

```
git push -u origin <nombre_de_la_rama>
```

Luego solo *git push* será suficiente para subir los cambios.

- **¿Qué es un repositorio remoto?**

El repo remoto es el que está en la nube (Github), el cual se puede ver desde cualquier lugar accediendo a la web (si tiene los permisos para verlo).

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

Para agregar un repositorio remoto a nuestro git (localmente) se debe usar el comando:

```
git remote add origin <url_del_repo>
```

También se puede clonar todo el repo con el comando:

```
git clone <url_del_repo>
```

La diferencia entre ambos básicamente es que el primero es solo un puntero al repositorio remoto (descarga info de ramas e historial de cambios pero no archivos), el segundo crea el puntero y además de descargar la info de ramas y commits también descarga los archivos del repo.

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

Como se explicó anteriormente, para subir los cambios al repo remoto se debe usar el comando:

```
git push -u origin <nombre_de_la_rama>
```

O simplemente:

```
git push
```

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

Para obtener los cambios desde un repo remoto se debe usar el comando:

```
git pull origin <nombre_de_la_rama>
```

O simplemente (si ya se ha hecho pull o push anteriormente):

```
git pull
```

- **¿Qué es un fork de repositorio?**

Un fork de un repositorio es hacer básicamente una copia de un repositorio de otro usuario en mi perfil, o sea crea un nuevo repositorio en mi cuenta con el código y visibilidad del otro repositorio.

Esto sirve si queremos modificar un proyecto pero no tenemos permisos de escritura. De esta forma podemos trabajar en el proyecto y adaptarlo o crear nuevas funcionalidades.

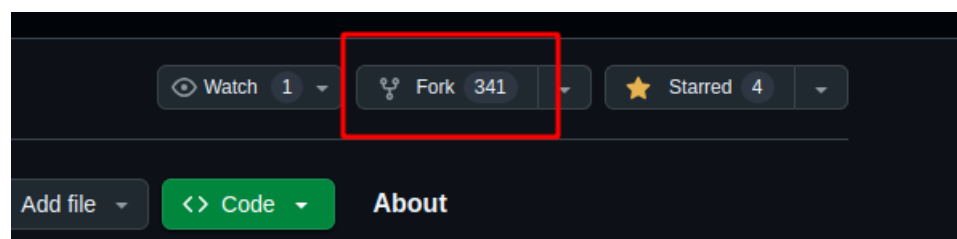
Es importante tener en cuenta que una vez hecho el fork no se va a actualizar con los nuevos cambios del otro proyecto. Ni mis cambios van a impactar en este.

Para mantener actualizado y hacer un pull request al original se deben hacer algunas configuraciones extras que se pueden [ver aquí](#).

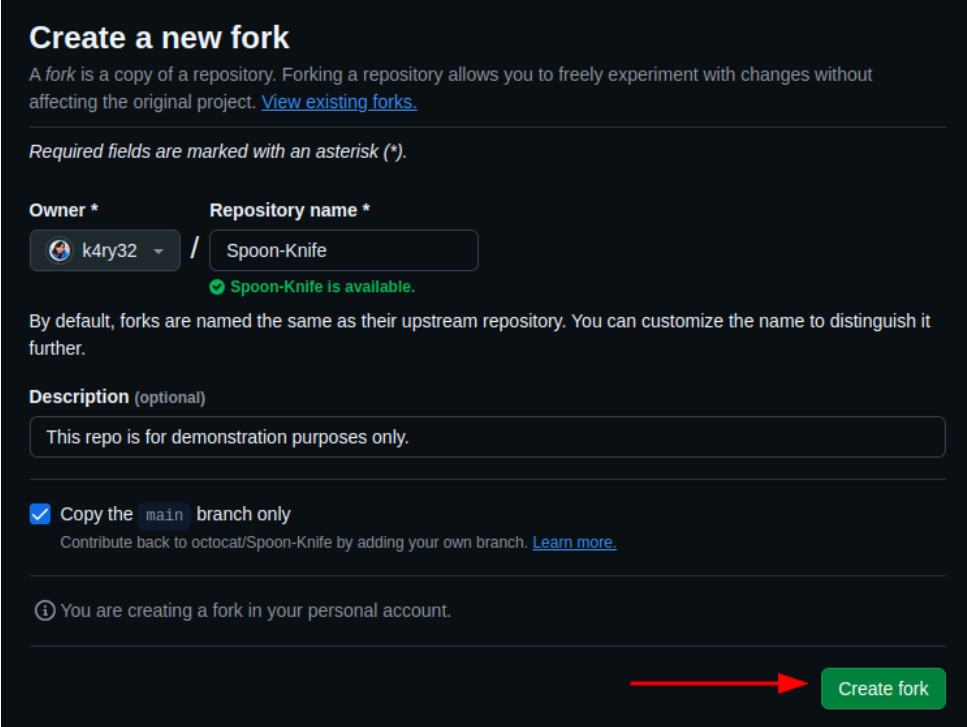
- **¿Cómo crear un fork de un repositorio?**

Para hacer un fork de otro repositorio hay que seguir los siguientes pasos:

1. Acceder al repositorio que desea hacer fork, y pulsar el botón "Fork".



2. Al presionar el botón se redirigirá a la pagina para preparar la nueva configuracion del fork:





Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk (*).

Owner * **Repository name ***

 k4ry32 / Spoon-Knife

 Spoon-Knife is available.


By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.


Description (optional)

This repo is for demonstration purposes only.

☒ Copy the `main` branch only

Contribute back to octocat/Spoon-Knife by adding your own branch. [Learn more.](#)

 You are creating a fork in your personal account.

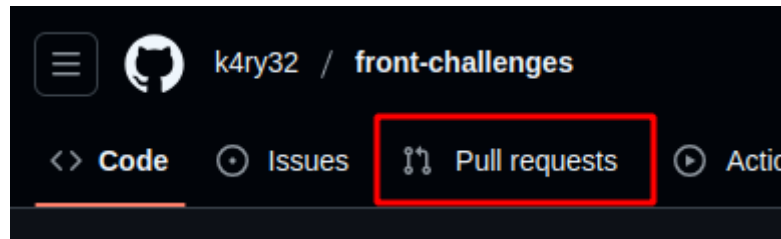
 **Create fork**

3. Aqui se podra cambiar el nombre si se desea al repositorio nuevo, agregar una descripción y elegir copiar solo la rama principal (por defecto) o todas si destildamos esa opción.
 4. Una vez lista la configuración presionamos en el botón “Create Fork”.
 5. Al hacer esto se redireccionará a tu nuevo repositorio creado a través del fork.
- **¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?**

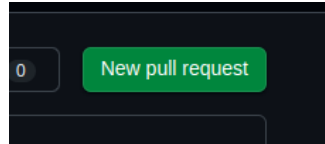
Luego de hacer cambios en nuestro local, hacer commit y push de la rama donde se trabajo hay que hacer un “pull request” para que revisen y acepten los cambios, si está todo ok, para incorporar los cambios a la rama principal o a la que se eligio para hacer la petición.

Para esto debemos:

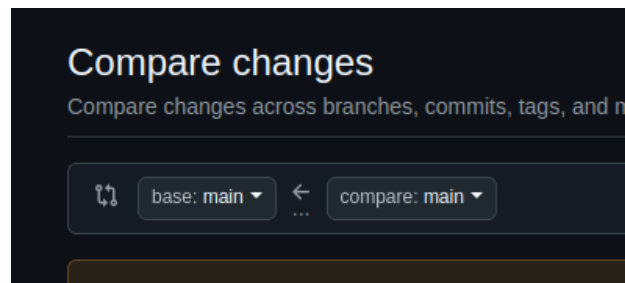
1. Ir a la página del repositorio en el que se desea realizar y seleccionar la pestaña “Pull requests”



2. Ahí presionar el botón “New pull request”



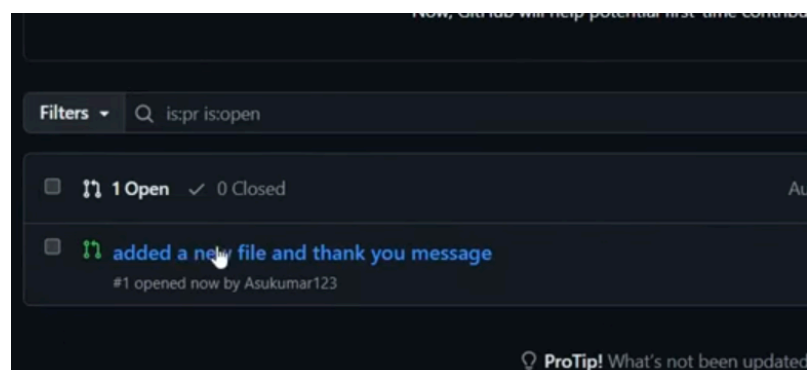
3. Luego se debe elegir la rama base a la cual se le van a aplicar los cambios (izquierda) y la rama que contiene los nuevos cambios (derecha):



4. Luego escribir un título y una descripción para la solicitud, por último dar al botón “Create pull request”.
 5. Después de crear una solicitud de cambios, puedes pedir a una persona concreta que revise los cambios propuestos.
- **¿Cómo aceptar una solicitud de extracción?**

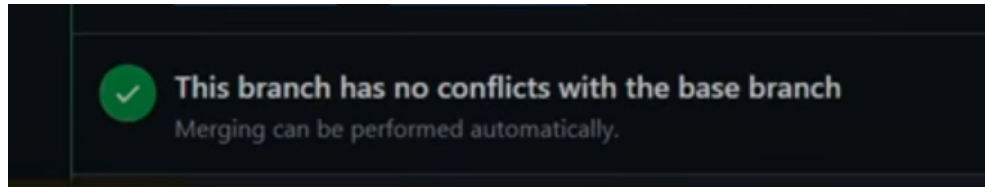
Se puede aceptar un pull request desde la página del repositorio, en la sección Pull requests. Al entrar ahí se verán las solicitudes (si hubiera alguna).

Se verá un listado parecido a este:



Al abrirlo se verá la descripción que se puso de los cambios hechos y se podrá comparar la rama base con la rama nueva para ver los cambios en detalle.

Lo principal para aceptar los cambios es que no debe haber conflictos con la rama principal, esto se verá con el siguiente cartel:



Si los cambios son correctos y no hay conflictos se puede aceptar la fusión con el botón “Merge pull request” y luego “Confirm merge”.

- **¿Qué es una etiqueta en Git?**

Las etiquetas en Git son referencias permanentes a puntos específicos en el historial del repositorio. A diferencia de los commits, que pueden moverse a medida que se hace git rebase y se reordena la historia del proyecto, las etiquetas son como marcadores permanentes que puedes usar para señalar versiones importantes, lanzamientos, o cualquier otro evento significativo en el desarrollo.

Mientras que las ramas en git son “colecciones” de commits, las etiquetas en git permiten referenciar un commit en concreto: marcarlo de forma inequívoca para poder encontrarlo de forma más rápida. Así mismo, mientras que las ramas permiten fusionarse con otras, una etiqueta en git es fija e inmutable además de apuntar siempre al mismo commit.

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

Hay dos tipos de etiquetas:

- Etiquetas Ligeras: Las etiquetas ligeras son simplemente referencias que apuntan a un commit específico. No contienen información adicional más allá del puntero al commit. Es algo así como ponerle un sobrenombre a un commit, de forma que pueda encontrar de forma más rápida.

Para crearla se utiliza el comando:

`git tag <nombre_etiqueta>`

- Etiquetas Anotadas: son objetos completos en Git que almacenan metadata adicional como el nombre del autor, la fecha, y un mensaje de anotación.

`git tag -a <nombre_etiqueta> -m "Mensaje descriptivo"`

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

Por defecto, git push no envía las etiquetas al origen salvo que se lo indiques explícitamente. Así que, hay dos opciones, enviar una etiqueta en concreto o bien, enviar todas las etiquetas. La primera opción es útil cuando solo quieres enviar una etiqueta al servidor remoto; la segunda es útil para sincronizar todas las etiquetas que tengas en tu máquina (y que el servidor no tenga).

```
# git push origin nombre-de-la-etiqueta
```

```
git push origin v1.2
```

```
# Enviar todas las etiquetas
```

```
git push origin --tags
```

- **¿Qué es un historial de Git?**

Es la lista de commits, en orden cronológico, que se han subido al repositorio. Contiene la información de que se cambió, quien lo realizó y cuando se subió.

- **¿Cómo ver el historial de Git?**

Mediante el comando git log podemos ver el historial de los últimos cambios o commits realizados en el proyecto.

Aparece desde el más reciente al más antiguo, y aparece paginado para que podamos movernos arriba y abajo, y salir pulsando q:

```
$ git log
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 10:31:28 2008 -0700

    first commit
```

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

Cuando necesitemos buscar algún mensaje específico de una confirmación anterior, tendremos que usar el siguiente comando:

```
git log --grep="Cat"
```

El comando anterior encontrará la palabra Cat en el historial de confirmaciones anteriores y mostrará los resultados al usuario con todas las confirmaciones coincidentes.

También se puede buscar por:

- Por cantidad: `git log -3`
- Por fecha: `git log --after="2014-7-1" --before="2014-7-4"`
- Por autor: `git log --author="John"`
- Por archivo: `git log -- foo.py bar.py`
- Por contenido: `git log -S"Hello, World!"`

Para saber más de filtrados [ver acá](#).

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

Una forma de borrar los commit hasta un punto deseado sería con el comando:

```
git reset <commit> --hard
```

Reemplazar <commit> con el hash del commit al que se desea volver. Hay que tener en cuenta que esto eliminará todos los cambios posteriores a ese commit.

Para borrar commit ya subidos al repositorio, luego del comando anterior hacer:

```
git push origin <branch> --force
```

Esto sobrescribirá el historial en el remoto.

Si solo se desea borrar el historial localmente, se puede eliminar la carpeta oculta .git en la raíz de tu proyecto. Esto eliminará todo el historial de Git, pero también se perderán todos los commits y configuraciones.

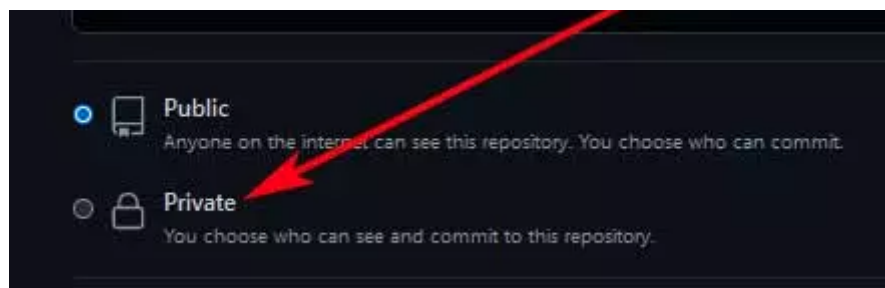
Para más formas de deshacer commits [ver aquí](#).

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

Es un repositorio al que solamente tiene acceso el propietario de la cuenta y creador del repositorio. Tampoco está visible al público.

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

Al momento de crear un nuevo repositorio hay una opción para configurarlo como público o privado:

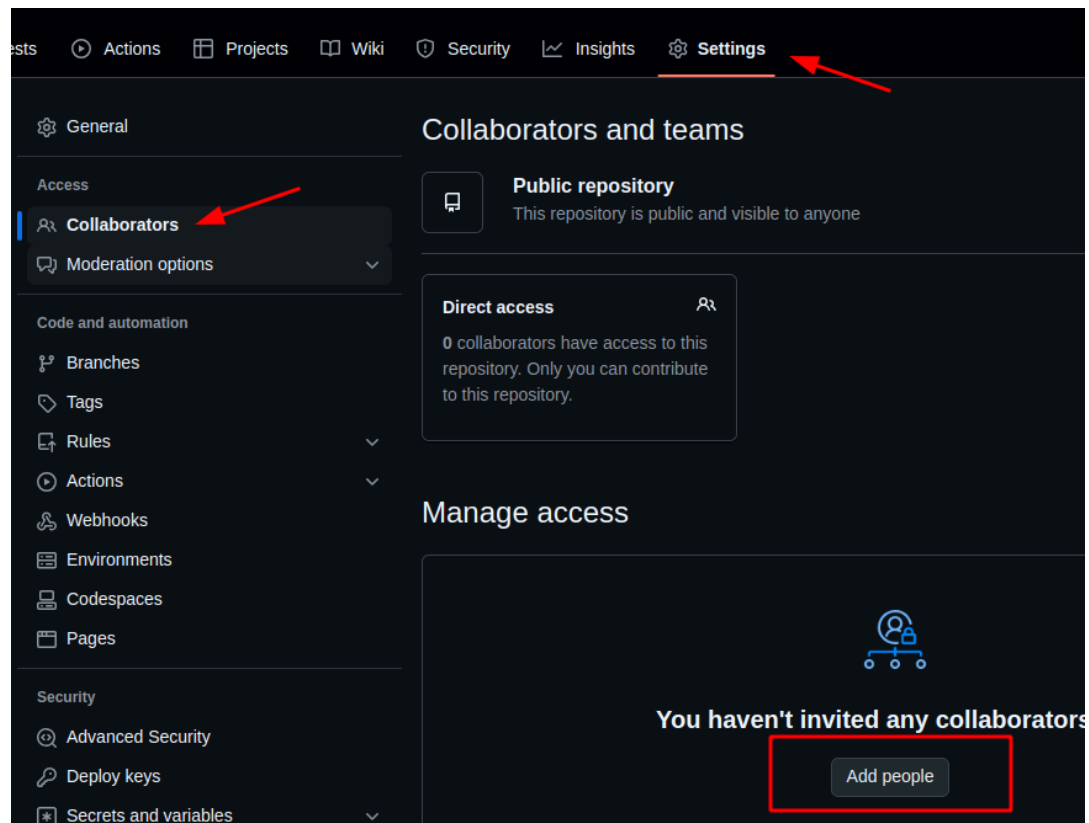


- **¿Cómo invitar a alguien a un repositorio privado en GitHub?**

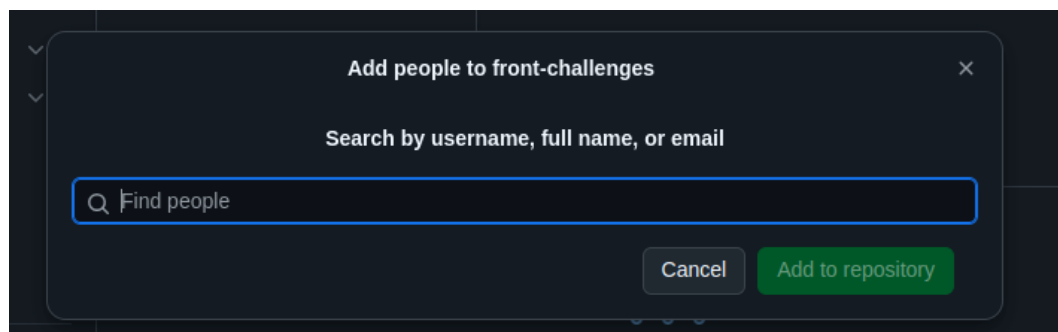
Para que otra persona pueda acceder al repositorio se puede enviar una invitación como colaborador del repositorio.

Para esto dirigirse al repositorio, ir a la pestaña “Settings” y luego entrar a “Collaborators”.

Desde aquí se podrá agregar a otro usuario con el botón “Add people”:



Se puede buscar por nombre de usuario, nombre completo o email con el que se creó la cuenta:



Finalmente, al elegir el usuario se agrega con el botón “Add to repository”.

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

Es un repositorio al que cualquier usuario tiene acceso a ver el código. También podrá hacer un fork del mismo.

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

Como se explicó anteriormente, se debe elegir la opción de repositorio público al momento de crearlo. Es la opción por defecto.

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

Con el enlace al repositorio simplemente ya se puede compartir.