

Rev.	Descripción de la Modificación			Fecha	Firma
00	Creación del documento			OCT'17	NOL
-	-				
PROYECTO N°	CANT.	OBSERVACIONES			PEDIDO EN PLANO N° POS.
			NOMBRE / NAME	FIRMA / INITIALS	FECHA / DATE
		PROYECTADO	Nahuel Olguin nahuelolguin98@gmail.com	NOL	OCT'17
		REVISADO			
		APROBADO			
<h1>GitHub</h1>					
<p>TITULO DEL DOCUMENTO:</p> <h2>Guía de trabajo en GitHub</h2>					
DOCUMENTO N°				REVISIÓN	HOJA

Contenido

1.	Organización	2
1.1	Crear organización	2
1.2	Invitación de miembros a la organización	3
1.3	Asignación de roles	5
1.4	Creación de equipos	6
2.	Repositorios	9
2.1	Crear y publicar repositorios.....	9
2.2	Subir cambios locales al repositorio remoto	13
3.	Modalidad de trabajo del Admin.....	16
3.1	Repositorios privados sin suscripción.....	16
4.	Modalidad de trabajo de los miembros	21
4.1	Clonar repositorio remoto en PC local.....	21
4.2	Modalidad de trabajo	23

1. Organización

1.1 Crear organización

Una vez que ya tenemos una cuenta personal de Github, para iniciar un repositorio en el que trabajaran un grupo de personas definidas es conveniente crear una organización asociada a nuestra cuenta, ya que nos ofrece herramientas que facilitan la coordinación del trabajo en equipo. Para ello en la esquina superior derecha nos vamos al selector con el símbolo “+” y creamos una nueva organización (Figura 1).

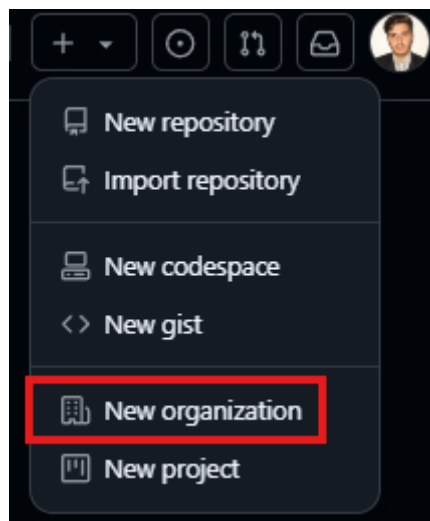


Figura 1: Creación de nueva organización

Nos aparecen diferentes planes de organización (Figura 2), cada uno con sus respectivas ventajas, pero por el momento podemos seleccionar la opción gratis (más adelante en el documento se mencionará una ventaja muy importante que ofrece la suscripción mensual).

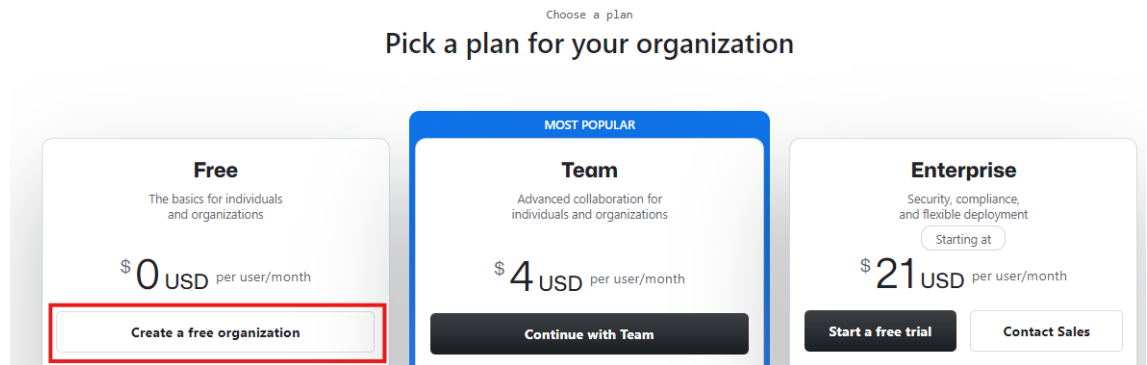


Figura 2: Planes de organización

Luego procedemos a asignar nombre, correo y seleccionar que mi cuenta personal esté vinculada a la organización (Figura 3). De esta forma queda finalizada la creación de nuestra organización

The image shows the 'Set up your organization' form on GitHub. The title is 'Set up your organization' with the subtitle 'Tell us about your organization'. There are two input fields: 'Organization name' (with a note: 'This will be the name of your account on GitHub. Your URL will be: https://github.com/') and 'Contact email'. Below these, there are two radio button options: 'My personal account' (selected, with the example 'I.e., Nahuel360 (Nahuel Olguin)') and 'A business or institution' (with the example 'For example: GitHub, Inc., Example Institute, American Red Cross').

Figura 3: Datos de organización

1.2 Invitación de miembros a la organización

Una vez finalizada la creación de la organización, nos vamos a la pestaña “people”, “members” y veremos que solo aparecemos nosotros como miembros de la organización. Para añadir nuevos miembros damos al botón “Invite member” (Figura 4).

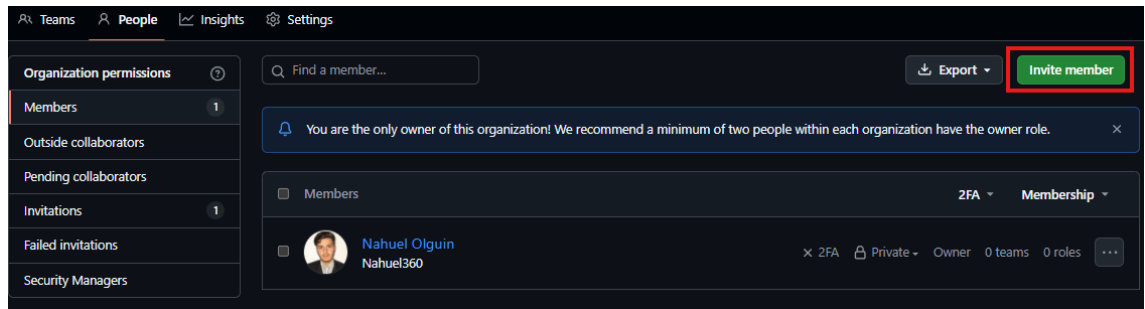


Figura 4: Invitación de nuevos miembros

Ingresamos el usuario que queremos añadir y seleccionamos el rol que le queremos asignar: “Member” si queremos que sea solo un miembro o “Owner” si queremos que sea un administrador con control total de la organización (Figura 5).

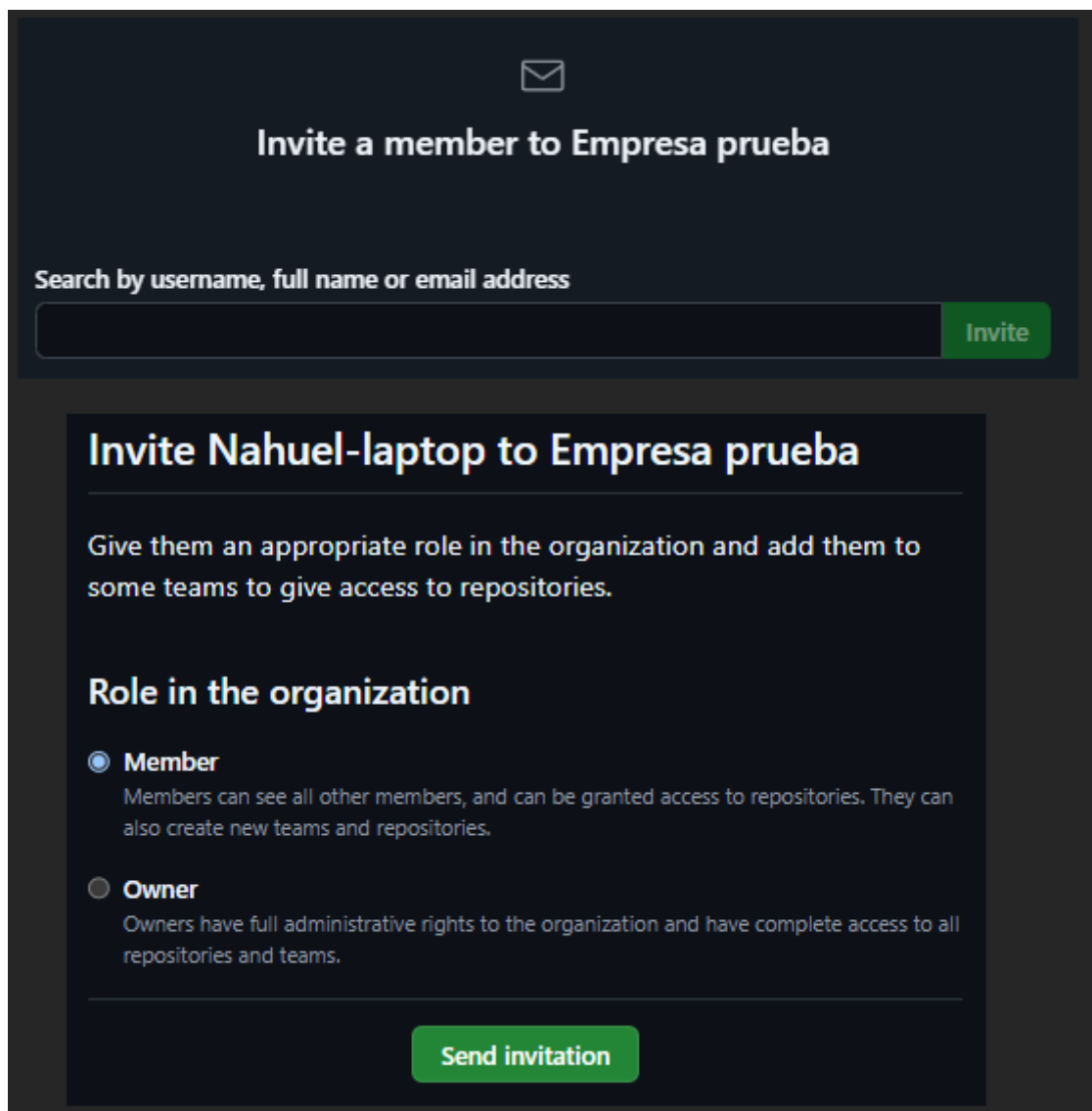


Figura 5: Proceso de invitación

De esta forma la otra persona recibirá un mail con la invitación para formar parte de la organización.

1.3 Asignación de roles

En la pestaña “setting” de la cuenta de la organización podemos ver la pestaña “organization roles” y “Role Management” ahí aparecen los roles que pueden ocupar los miembros dentro de la organización (Figura 6). Los dos más destacables son los roles de solo lectura (los miembros pueden ver, pero no modificar los repositorios) y los de escritura (los miembros pueden ver y modificar los repositorios).

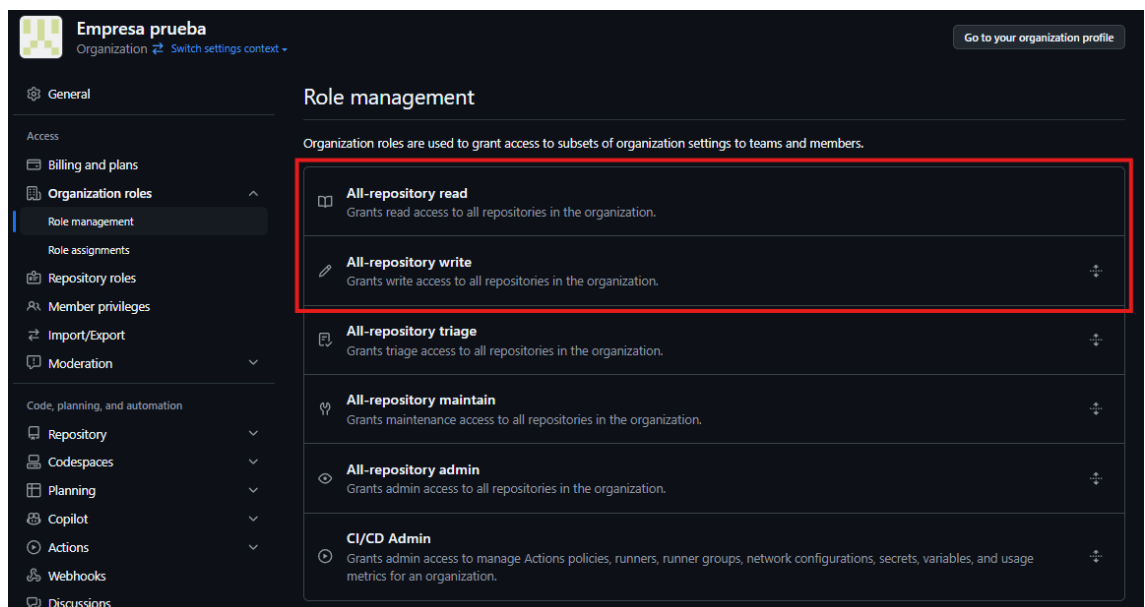


Figura 6: Roles dentro de la organización

Para asignar roles a cada miembro debemos ir a la pestaña “role assignments” y dar al botón “new role assignment” donde nos aparecerá una ventana donde debemos ingresar el nombre del miembro y seleccionar el rol que le vamos a otorgar (Figura 7).

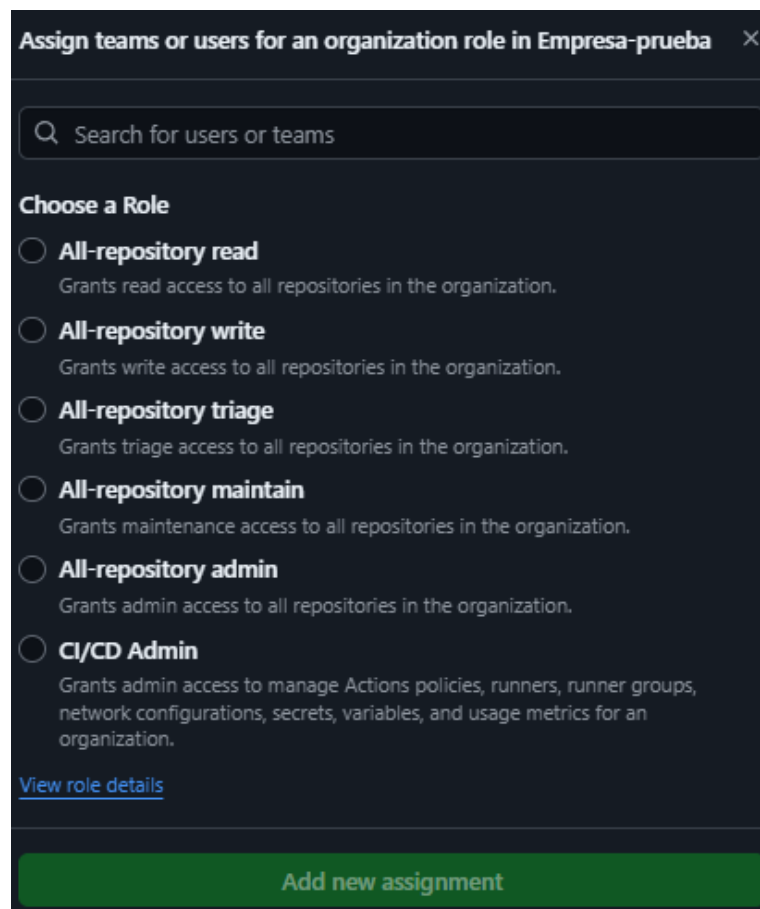
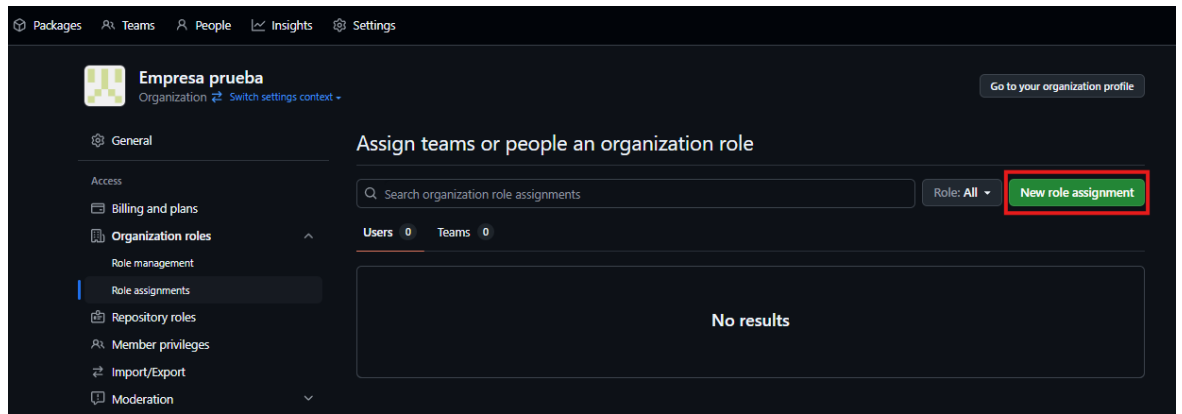
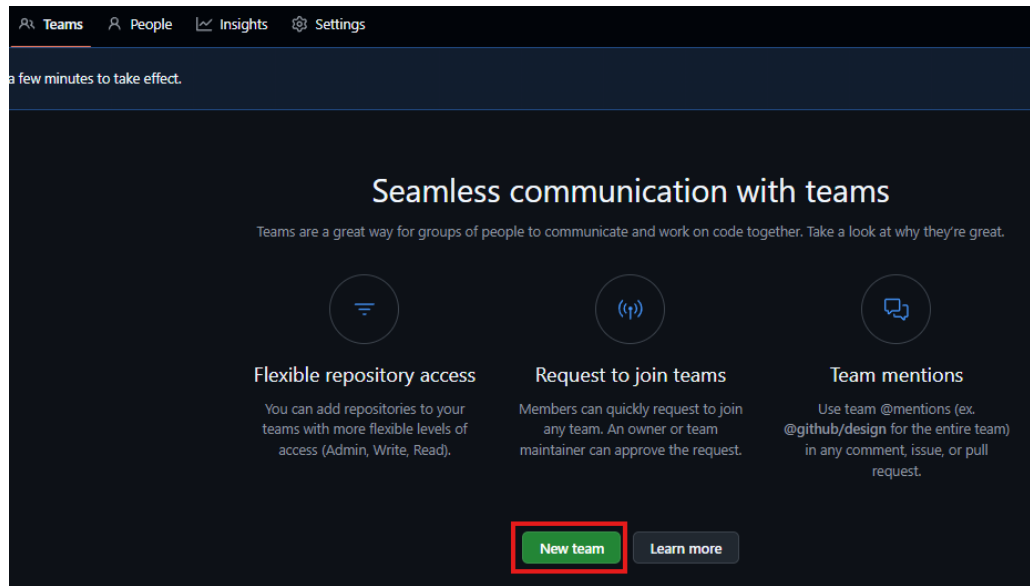


Figura 7: Asignación de roles

1.4 Creación de equipos

Si la organización cuenta con diferentes proyectos a realizar y queremos formar equipos de trabajo que tendrán acceso a repositorios específicos podemos crear equipos de trabajo. En la cuenta de la organización vamos a la pestaña “teams” y hacemos click en “New team”. Aparecerá la ventana para asignar nombre y descripción al grupo (Figura 8).



Create new team

Team name

You'll use this name to mention this team in conversations.

Description

What is this team all about?

Parent team

There are no teams that can be selected.

Team visibility

☒ **Visible** Recommended

A visible team can be seen and [@mentioned](#) by every member of this organization.

☐ **Secret**

A secret team can only be seen by its members and may not be nested.

Team notifications

☒ **Enabled**

Everyone will be notified when the team is @mentioned.

☐ **Disabled**

No one will receive notifications.

Create team

Figura 8: Creación de equipos

Una vez creado el equipo, añadimos los usuarios que formaran parte de este equipo. Finalmente nos vamos a repositorios y cargamos el repositorio de la organización con el que va a trabajar este equipo (Figura 9).

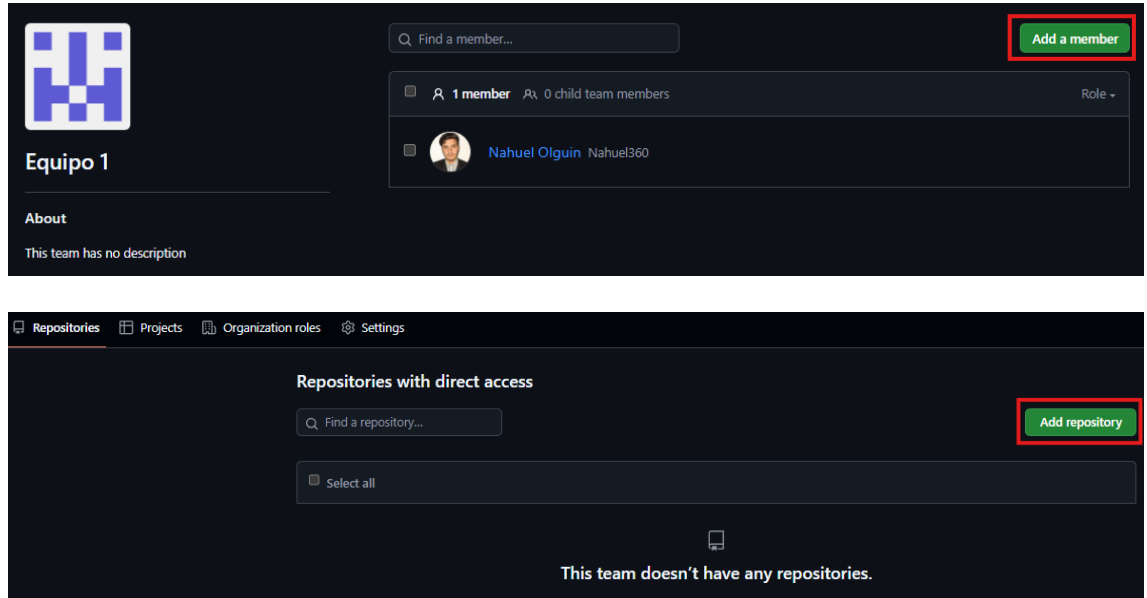


Figura 9: Añadir miembros y repositorios a un equipo

2. Repositorios

2.1 Crear y publicar repositorios

Si tenemos una carpeta en nuestra PC con la cual queremos dar origen a un nuevo repositorio, añadimos esa carpeta al “Github desktop” mediante la opción “Add local repository” (Figura 10).

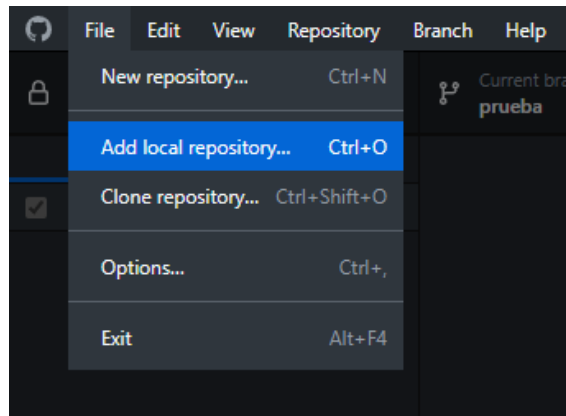


Figura 10: Añadir repositorio

Si la carpeta seleccionada no es un repositorio, nos muestra un mensaje que dice que la carpeta seleccionada no es un repositorio (Figura 11). Damos click en “create a repository” y nos aparece la ventana donde le asignamos el nombre y la descripción al repositorio.

Una vez creado el repositorio en nuestra maquina local le damos al botón “publish repository” para publicarlo en la web de Github y poder colaborar con otras personas de forma remota (Figura 12).

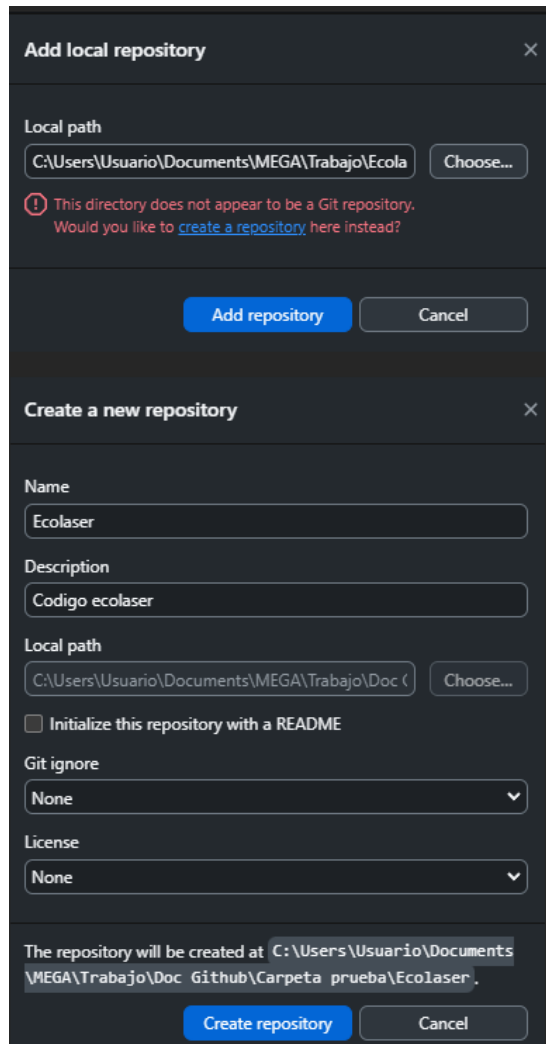


Figura 11: Creación de repositorio

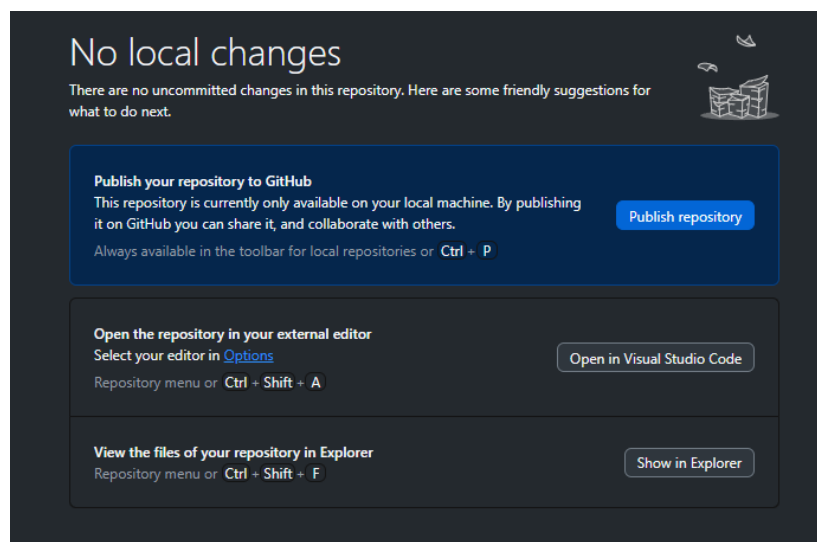
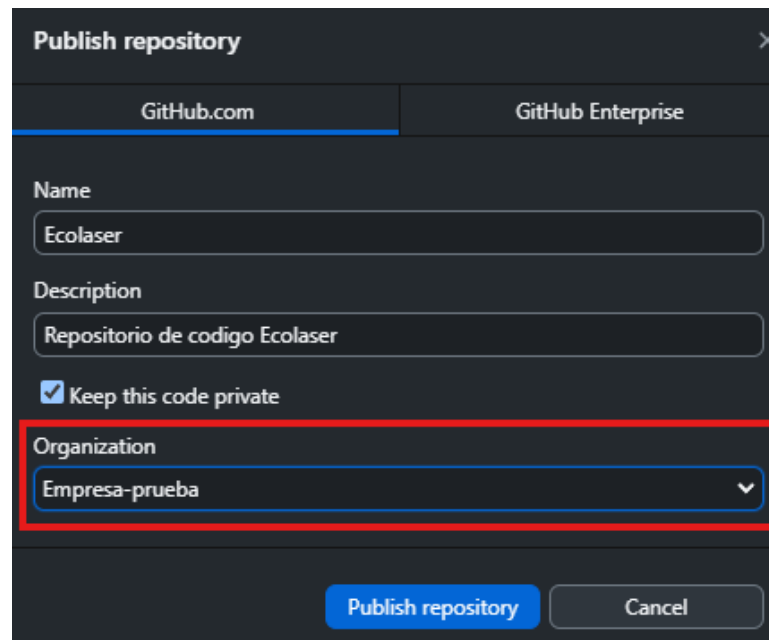


Figura 12: Publicación de repositorio

Aparece una ventana donde asignamos el nombre y descripción que queremos que aparezca en la web (Figura 13). En el selector “organization” podemos seleccionar la cuenta de la organización a la que pertenecerá el repositorio. Otra cosa importante a tener en cuenta es si vamos a decidir que el código sea público o privado con la opción “Keep this code private” ya que ambas presentan una serie de ventajas y desventajas:

- Privado: El repositorio solo podrá ser visualizado por nosotros y por personas a las que añadamos como miembros de la organización. La desventaja es que para hacer uso de ciertas herramientas tales como “Branch protection rules” que permiten gestionar mejor la limitación de las acciones de los miembros del repositorio, se debe pagar una suscripción mensual. Básicamente si no pagamos la suscripción, todos los miembros del repositorio tendrán acceso total al mismo, pudiendo realizar cambios sin la previa supervisión del admin, lo cual puede ser un problema si los integrantes no tienen conocimientos sólidos para trabajar de forma organizada.
- Publico: La desventaja es que el repositorio será visible para todo el mundo. Pero todas las funcionalidades de gestión mencionadas anteriormente están disponibles sin necesidad de pagar una suscripción.



Publish repository

GitHub.com | GitHub Enterprise

Name: Ecolaser

Description: Repositorio de codigo Ecolaser

☒ Keep this code private

Organization: Empresa-prueba

Publish repository | Cancel

Figura 13: Configuración del repositorio publicado

Una vez publicado el repositorio en Github, nos vamos a la web e ingresamos a nuestra organización donde nos aparecerá el repositorio que acabamos de subir y en la pestaña “code” se mostrarán todos los archivos que estaban originalmente en la carpeta local (Figura 14).

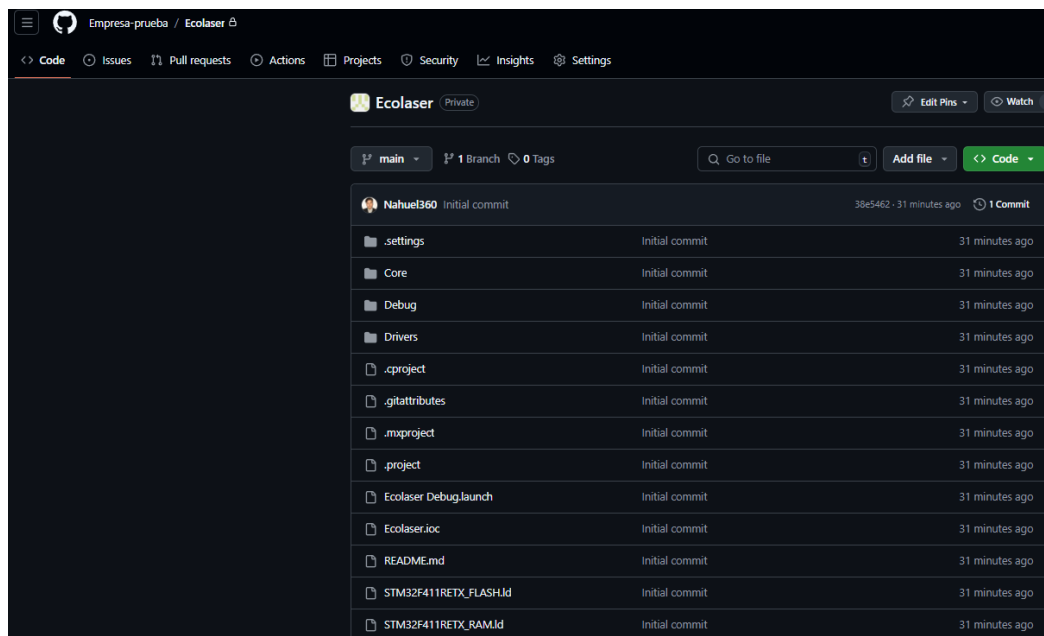


Figura 14: Repositorio cargado en GitHub

2.2 Subir cambios locales al repositorio remoto

Ahora cualquier cambio que realicemos en el código local podremos subirlo al repositorio de la web para que todos los integrantes del equipo de trabajo puedan visualizarlo. Por ejemplo, vamos modificar el código en nuestra máquina local, agregando un comentario de saludo “Hola soy el programador local” (Figura 15).

```
19  /* Includes ----- */
20  //Hola soy el programador local
21  #include "main.h"
```

Figura 15: Modificación de código local

Al guardar los cambios en el editor de código que se esté usando (puede ser cualquier editor), veremos en el Github desktop que hemos realizado un cambio en un archivo y nos muestra dicho cambio (Figura 16).

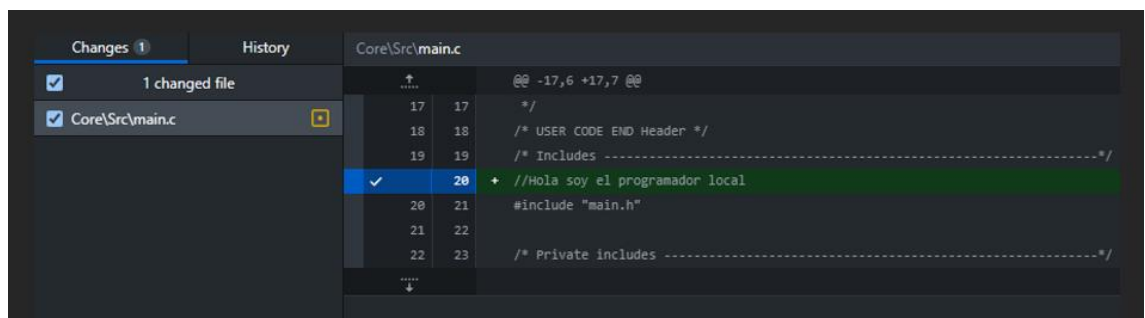


Figura 16: Visualización de cambios en GitHub Desktop

En la esquina inferior izquierda, hay un botón etiquetado como “Commit to main”, que se utiliza para confirmar y guardar los cambios en el repositorio local (Figura 17). Previamente se deben llenar dos campos:

- Summary (required): Aquí debes ingresar un resumen obligatorio de los cambios que estás realizando.
- Description: Este campo es opcional y puedes usarlo para proporcionar una descripción más detallada de los cambios.

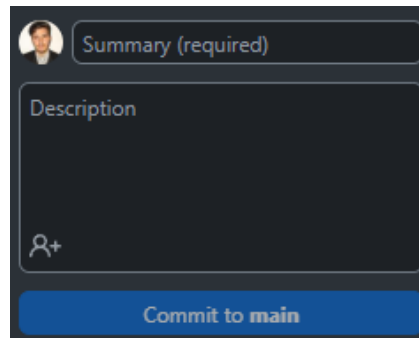


Figura 17: Guardar cambios en repositorio local

Finalmente, para cargar dichos cambios en el repositorio remoto de Github web se debe hacer el push del commit dando al botón “Push origin” (Figura 18).

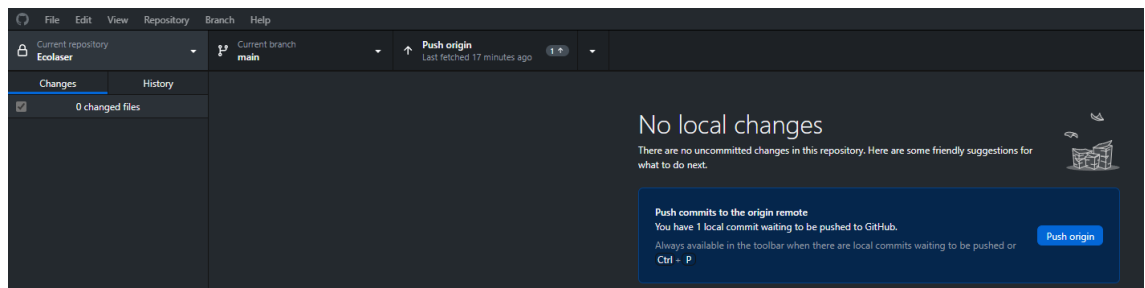


Figura 18: Guardar cambios en repositorio remoto

Ahora si nos vamos al repositorio en Github web y abrimos el código en el que se realizaron los cambios, vemos que aparece el cambio realizado de forma local (Figura 19).

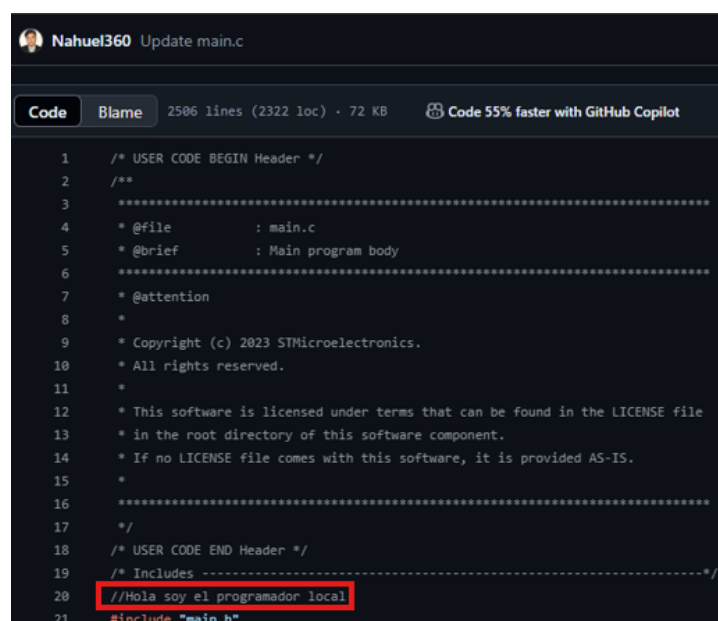


Figura 19: Archivo modificado en repositorio local

En Github desktop en la pestaña history podemos ver todos los cambios realizados, cuando se realizaron y quien los realizo (Figura 20).

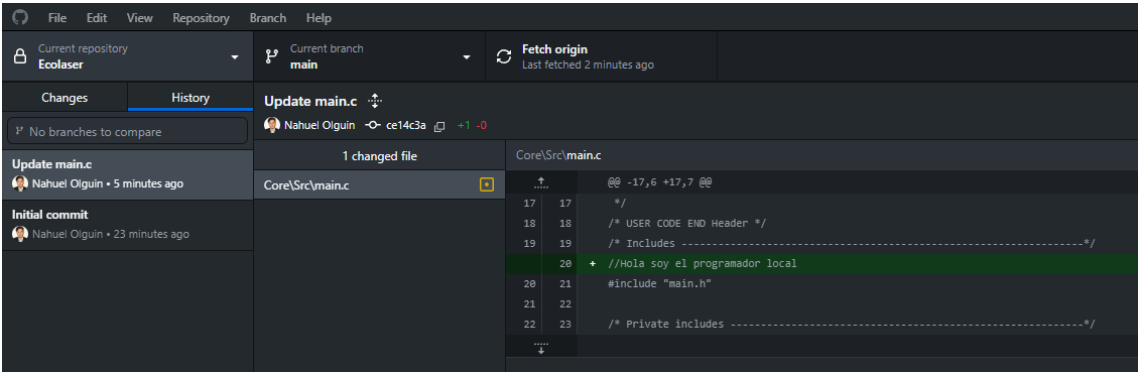


Figura 20: Historial de cambios realizados

3. Modalidad de trabajo del Admin

3.1 Repositorios privados sin suscripción

Una forma eficiente de trabajar sobre un repositorio es el modelo de ramas (Branching Model), en el cual cada persona del equipo cuente con su propia rama de trabajo y al finalizar su parte, esta rama vuelva a fusionarse a la rama principal del proyecto. Este flujo de trabajo ayuda a mantener el código organizado, facilita la colaboración y reduce el riesgo de conflictos y errores.

Como se mencionó anteriormente, cuando queremos trabajar en equipo en un repositorio privado sin contar con un plan de pago, no contamos con las “Branch protection rules” las cuales permiten asignar restricciones de acceso a los miembros a las ramas de un proyecto. Por este motivo es importante que todos los integrantes del proyecto conozcan la modalidad de trabajo para evitar errores y conflictos.

Tareas que debe realizar el admin en la metodología Branching Model:

1. Crear rama: se debe crear una rama por cada tarea y asignarla únicamente a un solo equipo o miembro. Para crear una rama debemos ingresar al menú “Branch” del repositorio, dentro de ese menú le damos click a “New Branch” y en la ventana emergente le asignamos el nombre a la rama (Figura 21). El nombre debe ser acorde a la tarea a realizar, por ejemplo, si ese equipo se debe encargar de implementar un control de temperatura en el código, el nombre de la rama podría ser “temperatura”.

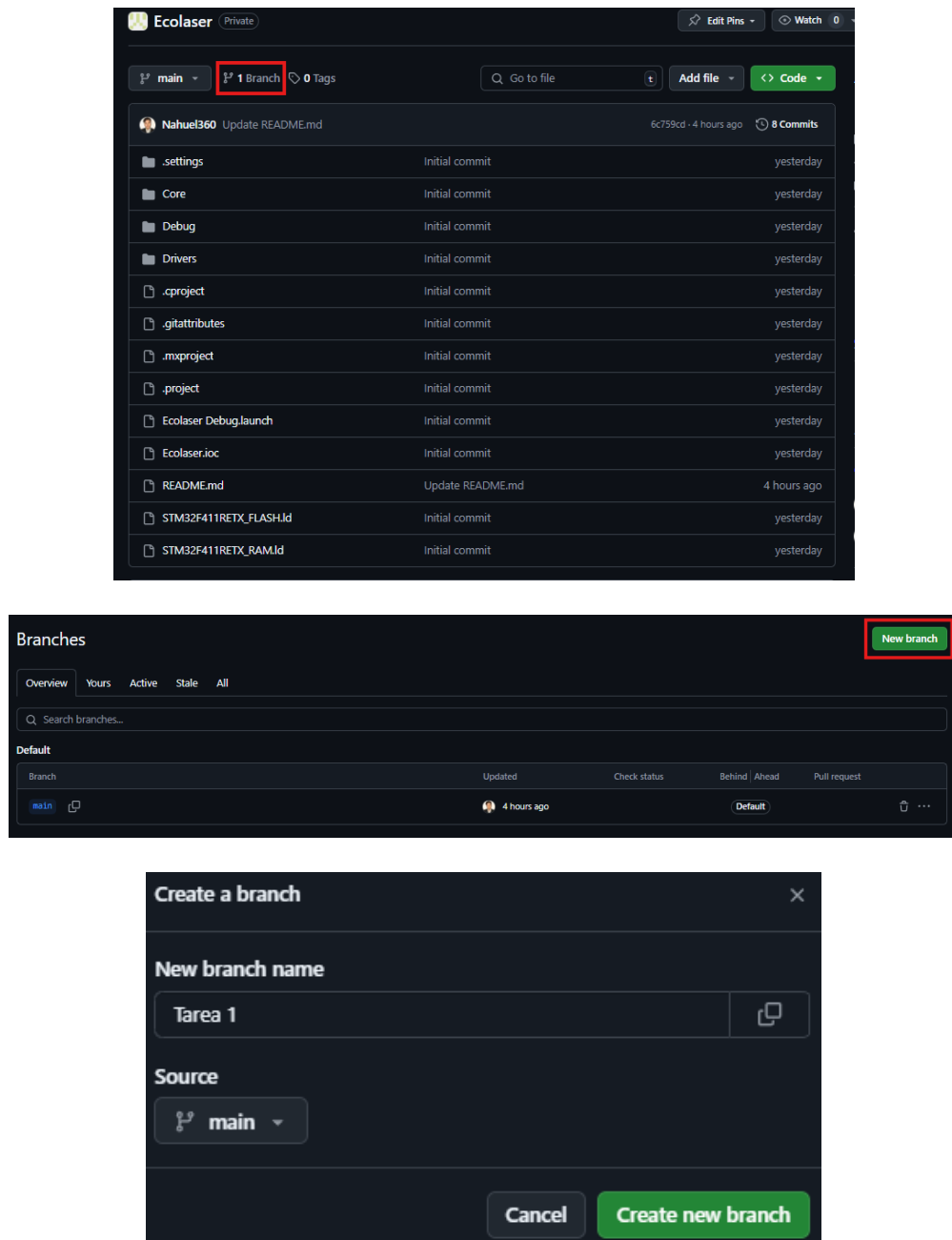
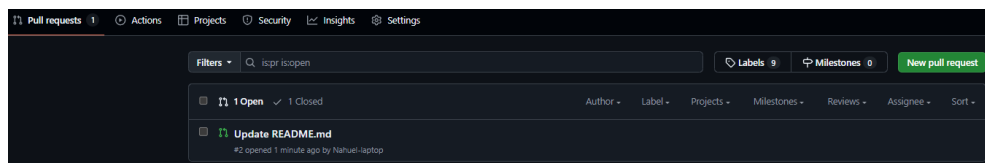


Figura 21: Creacion de ramas

2. Asignar la rama: Una vez creada la rama se debe comunicar claramente cual equipo o miembro de la organización trabajara en esa rama (Aclarar que el resto de equipos o miembros tienen prohibido realizar cambios en esta rama).

3. Verificar Pull request: Cuando el equipo o miembro que trabajaba sobre esa rama considera que finalizo su tarea, enviaron una solicitud o “pull request” para que los cambios sean revisados por el admin.

Para poder verlas debemos entrar a la pestaña “Pull requests” del repositorio y se observarán las mismas, en la siguiente imagen se observa una correspondiente a la actualización del archivo readme llamada “Update README.md”.



Al abrir la pull request se nos dan las siguientes opciones:

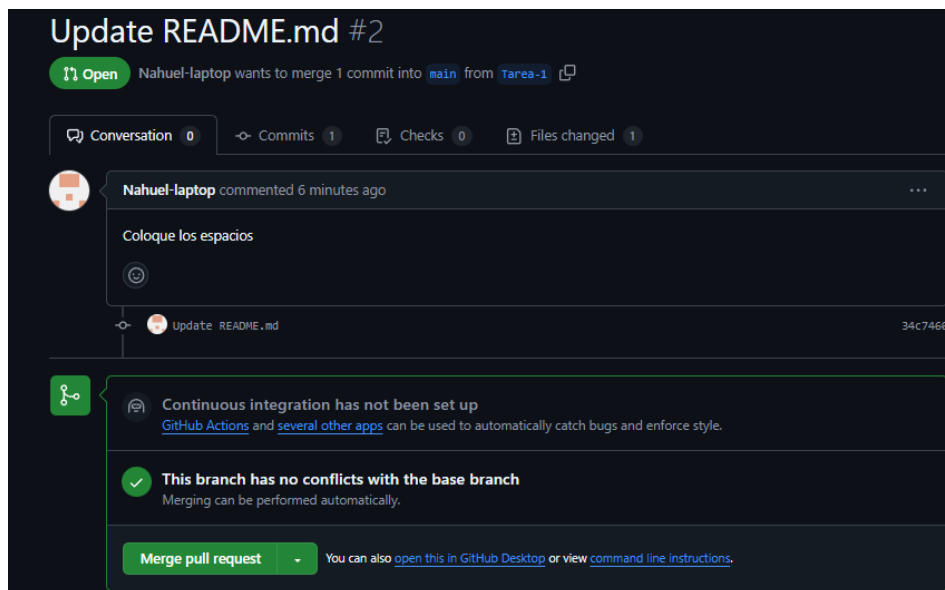
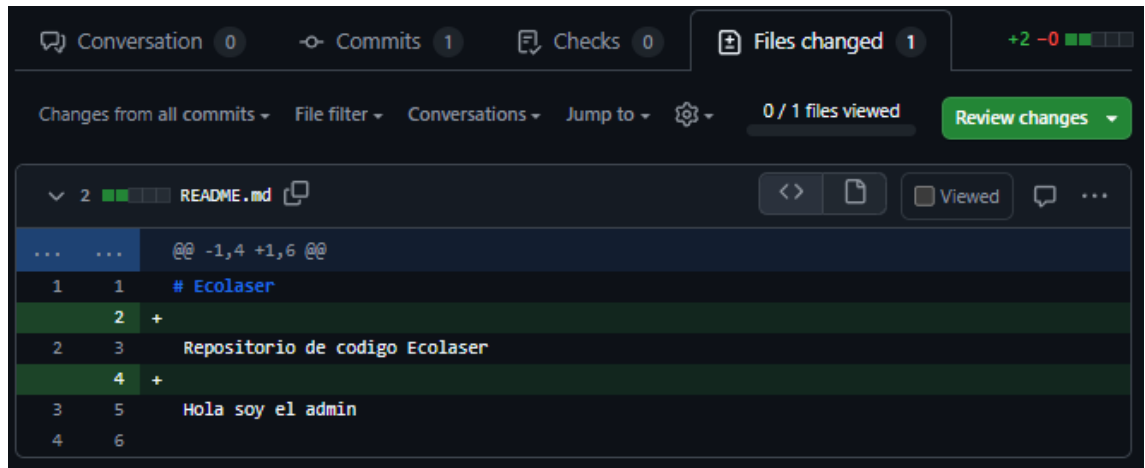
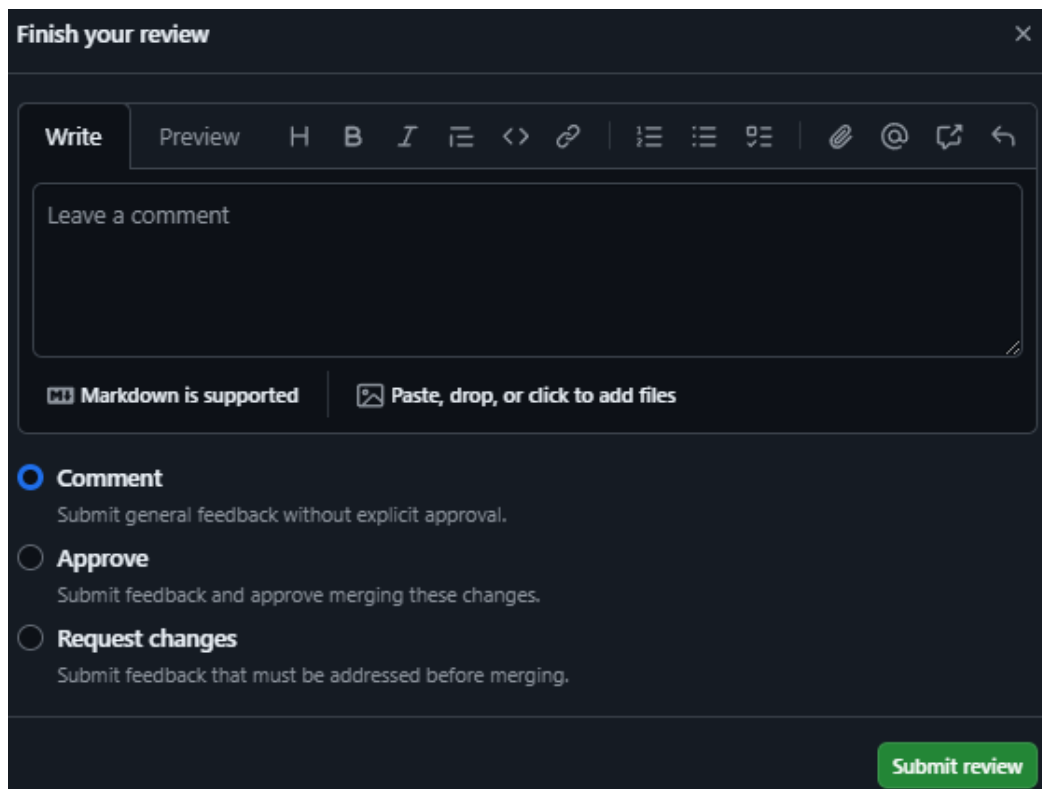


Figura 22: Menú de Pull request

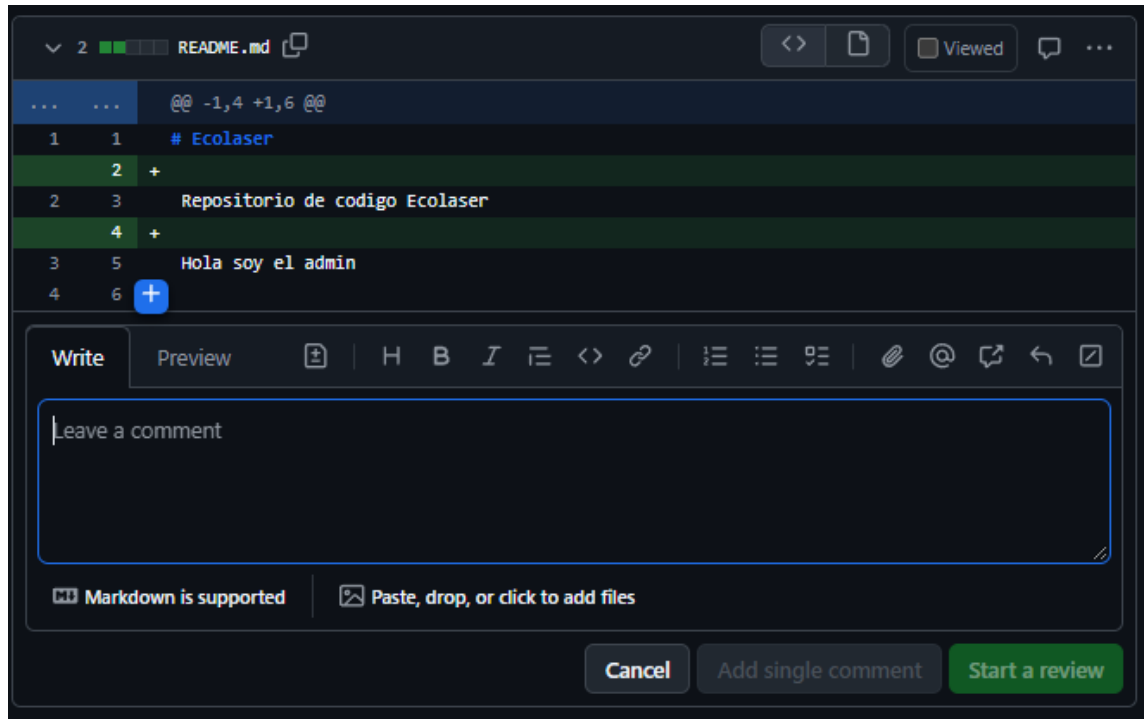
Lo primero que se debe hacer es ingresar en la pestaña “files changed” para ver los cambios propuestos en el código.



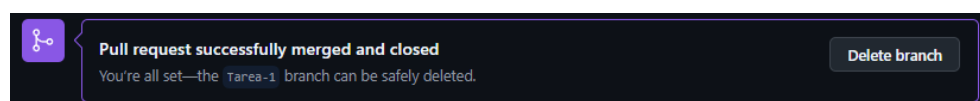
Si damos click a “Review changes” nos aparecerá una ventana donde podemos dejar tres tipos de mensajes generales: comentario neutro (comment), mensaje de aprobación (Approve) o solicitar cambios (Request changes). Estos mensajes quedarán guardados en la pestaña “conversation” para que los miembros que realizaron la pull request puedan visualizarlo.



Además, si pasamos por encima de las líneas de código aparecen símbolos “+” que nos permiten dejar comentarios en líneas de código específicas. Esto es útil para comunicar correcciones que se deban realizar al código.



4. Fusionar la rama: Una vez que el admin considera que el Pull Request ha sido aprobado, se puede fusionar la rama de trabajo con la rama principal dando click a “Merge Pull request” (Figura 22) y posteriormente “Confirm merge”, esto actualiza el código principal con los nuevos cambios (cuando hay más de una rama trabajando en paralelo es común observar que al momento de querer unirlos Github nos comuniquen que se presentan conflictos entre ramas y nos ofrezca una interfaz para resolverlos).
5. Eliminar la rama: Después de fusionar, es una buena práctica eliminar la rama de trabajo para mantener el repositorio limpio y organizado.



4. Modalidad de trabajo de los miembros

4.1 Clonar repositorio remoto en PC local

Desde el otro pc nos llegara un mail con la invitación para participar del proyecto, hacemos click y aceptamos la invitación (Figura 23).

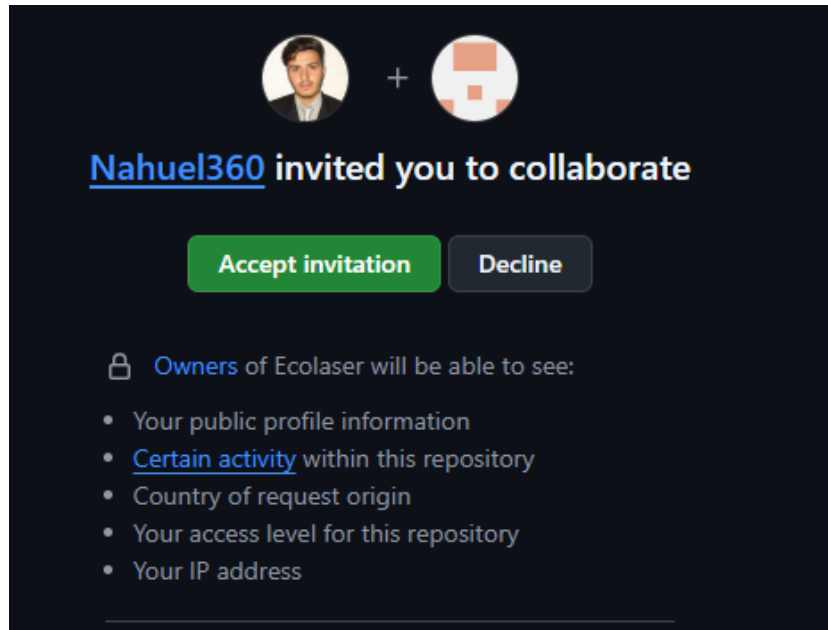


Figura 23: Invitación para colaborar en el proyecto

Ahora podemos acceder al Github desktop y clonar en nuestro pc local todos los repositorios remotos a los que el admin nos otorgue acceso. Antes de clonar, se debe elegir la ubicación "Local path" en donde se guardarán todos los archivos (Figura 24).

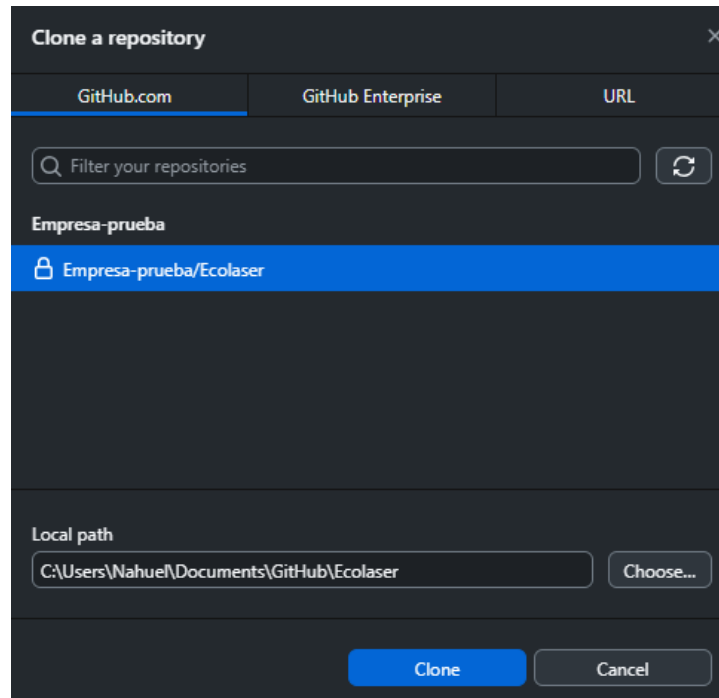


Figura 24: Clonación de repositorio

En la ubicación del “Local path” seleccionado encontraremos todos los archivos correspondientes al repositorio y podremos subir las modificaciones al repositorio remoto con el mismo procedimiento visto para el admin en la sección “2.2”.

Como miembro es común que a veces aparezca una advertencia cuando se quiere hacer un commit que indica que no tienes permisos de escritura en el repositorio (Figura 25). Esto significa que no puedes hacer cambios directamente en ese repositorio, hasta que el admin te otorgue los permisos de escritura como se explica en la sección “1.3”.

Nota: La opción que ofrece es crear un “fork”, que es una copia personal del repositorio donde puedes hacer cambios sin afectar el original ya que se trata de un proyecto completamente ajeno. Sin embargo, esta opción generalmente está bloqueada por el admin por defecto y no es de nuestro interés por el momento.

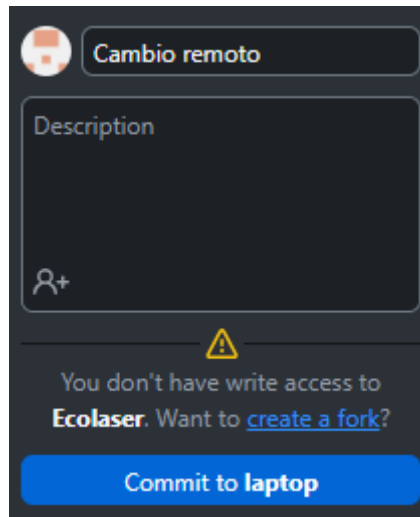


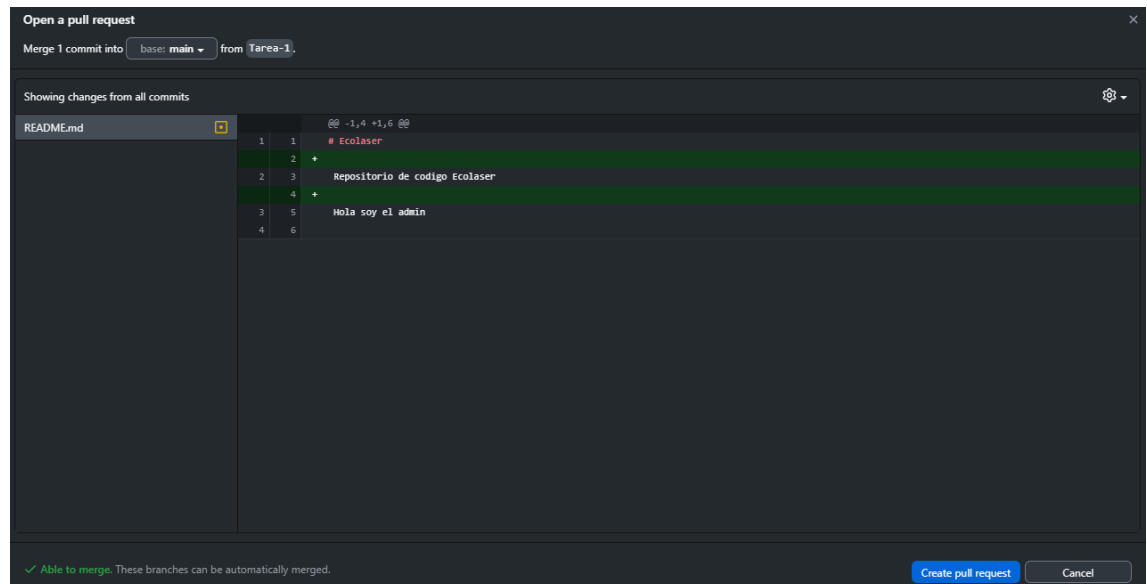
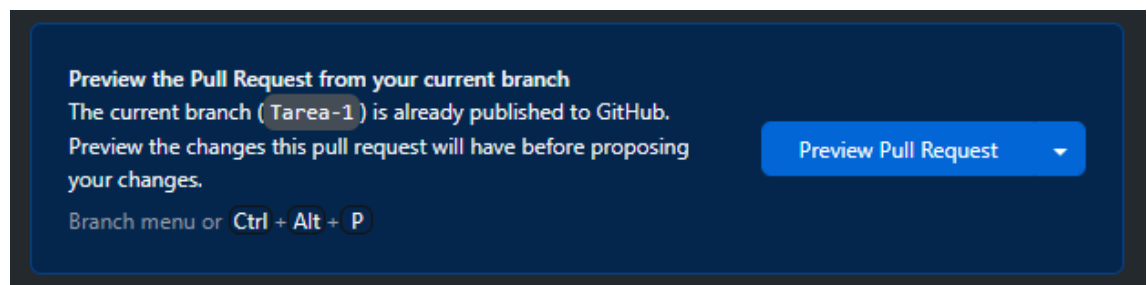
Figura 25: Advertencia por falta de permisos

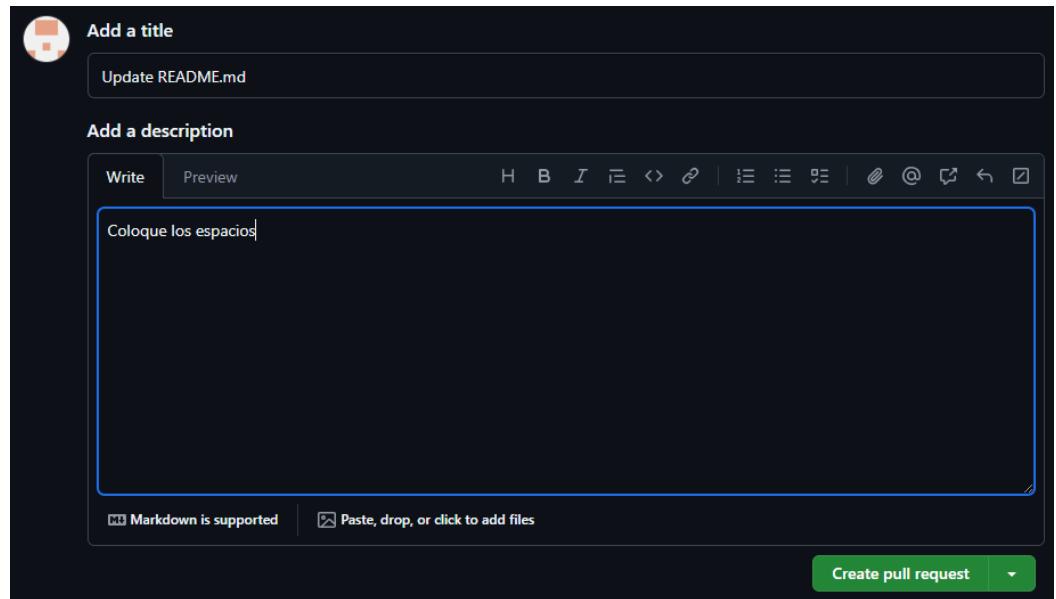
4.2 Modalidad de trabajo

En la sección “3.1” se explica que es la metodología de trabajo Branching Model y la importancia de conocer el procedimiento (recomendamos leer completamente esa sección). Se proceden a explicar las tareas que debemos realizar como miembros de una organización en la metodología Branching Model:

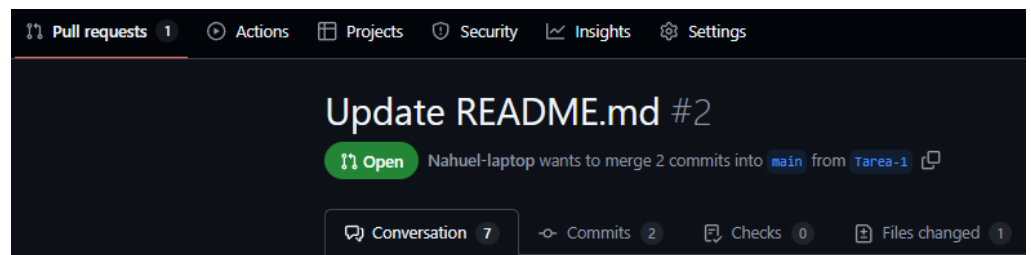
1. El admin nos asignara una tarea con su respectiva rama a nosotros o a nuestro equipo. Es muy importante que trabajemos únicamente solo en las ramas que nos asignaros (está prohibido realizar cambios en ramas a las que no fuimos asignados). Esto permite que cada persona desarrolle y pruebe sus cambios de manera aislada sin afectar el código principal o ramas de otros.
2. **Realizar commits:** Hacer commits en nuestra rama de trabajo para mantener un historial claro y detallado de los cambios realizados (verificar que estemos siempre en la rama correcta antes de realizar commits).
3. **Realizar push de commits:** subir los cambios al repositorio remoto de Github web.

4. **Revisar y probar los cambios:** Antes de solicitar un pull request de la rama de trabajo con la rama principal, es importante revisar y probar los cambios de forma local para asegurarse de que no introduzcan errores.
5. **Pull Request (PR):** Crear un Pull Request para fusionar la rama de trabajo con la rama principal. Esto permite que el admin revise los cambios y sugiera mejoras o correcciones en caso de ser necesario. Dar a “Preview Pull request” para visualizar los cambios antes de crear la solicitud de revision y si esta todo correcto realizar la pull request dando click a “create pull request”. Nos enviara a la web donde debemos dar título y descripción a la pull request antes de crearla.



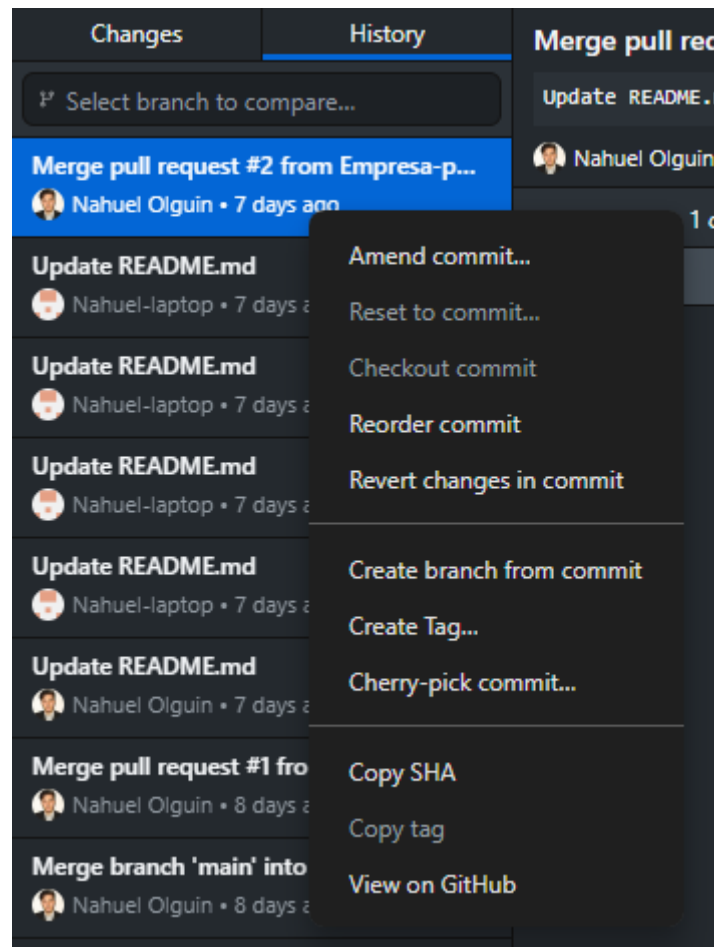


6. Estar pendiente de la pestaña “Conversation” del pull request para cuando el admin ofrezca una de las siguientes respuestas:
- Solicite correcciones o cambios: en este caso debemos realizar las correcciones necesarias en el código y realizar commits y pushes nuevamente.
 - Apruebe el trabajo: en este caso nuestra tarea en la rama de trabajo se da por finalizado.



4.3 Funciones importantes

Si vamos a la pestaña history nos aparecen todos los commits realizados y si hacemos click derecho en uno de ellos nos muestra las siguientes opciones:



Se procede a dar una explicación de las opciones que aparecen en GitHub

Desktop:

1. **Amend commit:** Permite modificar el último commit realizado. Puedes cambiar el mensaje del commit o agregar nuevos cambios antes de confirmar.
2. **Reset to commit:** Restablece el repositorio al estado de un commit específico (Podemos volver a un commit anterior siempre y cuando no hayamos realizado ningún push al repositorio web posterior a ese commit). Esto puede eliminar los cambios realizados después de ese commit.
3. **Checkout commit:** Cambia el estado del repositorio al de un commit específico. Esto te permite ver cómo estaba el proyecto en ese momento, pero estarás en un estado “detached HEAD” (No pertenecemos a ninguna rama del proyecto).

4. **Revert changes in commit:** Crea un nuevo commit que deshace los cambios realizados en un commit específico.
5. **Create branch from commit:** Crea una nueva rama a partir de un commit específico. Esto es útil si deseas desarrollar una nueva funcionalidad basada en el estado del proyecto en ese commit.
6. **Create Tag:** Asigna una etiqueta a un commit específico. Las etiquetas son útiles para marcar versiones importantes del proyecto.
7. **Cherry-pick commit:** Aplica los cambios de un commit específico a la rama actual. Esto es útil para traer cambios específicos sin fusionar toda la rama.
8. **Copy SHA:** Copia el hash del commit al portapapeles. El hash es un identificador único para cada commit.