

Alumno: Santiago Raúl Salinas

Materia: Programación I

Práctico 2: Git y GitHub

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

- ¿Qué es GitHub?
 - ¿Cómo crear un repositorio en GitHub?
 - ¿Cómo crear una rama en Git?
 - ¿Cómo cambiar a una rama en Git?
 - ¿Cómo fusionar ramas en Git?
 - ¿Cómo crear un commit en Git?
 - ¿Cómo enviar un commit a GitHub?
 - ¿Qué es un repositorio remoto?
 - ¿Cómo agregar un repositorio remoto a Git?
 - ¿Cómo empujar cambios a un repositorio remoto?
 - ¿Cómo tirar de cambios de un repositorio remoto?
 - ¿Qué es un fork de repositorio?
 - ¿Cómo crear un fork de un repositorio?
 - ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
 - ¿Cómo aceptar una solicitud de extracción?
 - ¿Qué es un etiqueta en Git?
 - ¿Cómo crear una etiqueta en Git?
 - ¿Cómo enviar una etiqueta a GitHub?
 - ¿Qué es un historial de Git?
 - ¿Cómo ver el historial de Git?
 - ¿Cómo buscar en el historial de Git?
 - ¿Cómo borrar el historial de Git?
 - ¿Qué es un repositorio privado en GitHub?
 - ¿Cómo crear un repositorio privado en GitHub?
 - ¿Cómo invitar a alguien a un repositorio privado en GitHub?
 - ¿Qué es un repositorio público en GitHub?
 - ¿Cómo crear un repositorio público en GitHub?
 - ¿Cómo compartir un repositorio público en GitHub?
-
- ¿Qué es GitHub?

GitHub es una plataforma de desarrollo colaborativo que permite a los desarrolladores almacenar, gestionar y compartir código fuente de manera eficiente.

Principales funciones:

- Control de versiones
- Colaboración
- Alojamiento de código
- Gestión de proyectos
- Comunidad

- ¿Cómo crear un repositorio en GitHub?

Primero debes estar registrado en el sitio Web de GitHub.

Una vez que estes registrado, vas a poder configurar tu perfil y crear un repositorio o varios.

Para crear un repositorio en GitHub,

1. Ve a [GitHub](#)
2. En la esquina superior derecha, selecciona Nuevo repositorio
3. Escribe un nombre para el repositorio
4. Agrega una descripción opcional
5. Selecciona la visibilidad del repositorio
6. Haz clic en Crear repositorio

- ¿Cómo crear una rama en Git?

Para crear una nueva rama, se usará el comando git branch:

```
$ git branch "NombredelaRamaNueva"
```

Git sabe en qué rama estás mediante un apuntador especial denominado HEAD. Por más que creemos una nueva rama permanecemos en la rama principal o por defecto llamada main o master.

- ¿Cómo cambiar a una rama en Git?

Para moverse de una rama a otra, tienes que utilizar el comando git checkout:

```
$ git checkout "NombredelaRamaNueva"
```

En este caso si se traslada el apuntador HEAD a la rama especificada.

Para crear una nueva rama y saltar a ella, en un solo paso, puedes utilizar el comando git checkout con la opción -b:

```
$ git checkout -b "NombredelaRamaNueva"
```

- ¿Cómo fusionar ramas en Git?

Para fusionar ramas en Git, sigues un proceso que combina los cambios de una rama en otra.

Primero, debes estar en la rama a la que quieres fusionar los cambios. Por lo general, es a la rama principal (master o main) o cualquier otra rama en la que estés integrando los cambios.

Pongamos de ejemplo que queremos fusionar la rama

"NombredelaRamaNueva" a la rama master, entonces vamos a pararnos en la principal:

```
$ git checkout master
```

Una vez en la rama de destino, utilizamos el comando git merge para fusionar la rama de origen en la rama actual:

```
$ git merge "NombredelaRamaNueva"
```

Este comando incorpora los cambios de "NombredelaRamaNueva" en master.

- ¿Cómo crear un commit en Git?

Crear un commit en Git es el proceso mediante el cual se guardan los cambios realizados en el repositorio. Un commit captura el estado actual del código en un momento dado y lo almacena en el historial del proyecto.

Pasos para hacer un commit:

Primero realiza los cambios en los archivos de tu proyecto que desees guardar en el commit.

Luego debes agregar los cambios al área de preparación. Esto se hace con el comando git add. Puedes agregar archivos específicos o todos los archivos modificados:

```
$ git add "nombredelarchivo" (para agregar las modificaciones de ese archivo en específico)
```

```
$ git add . (para agregar las modificaciones de todos los archivos en general)
```

Una vez que los cambios están en el área de preparación, puede hacer el commit con el comando git commit. Debes proporcionar un mensaje de commit que describa los cambios realizados:

```
$ git commit -m "Mensaje de los cambios"
```

Esto guarda el estado actual del código en el historial del repositorio con el mensaje correspondiente.

- ¿Cómo enviar un commit a GitHub?

Primero debemos clonar el repositorio de GitHub a nuestra máquina local:

```
$ git clone https://github.com/tu_usuario/tu_repositorio.git
```

```
$ cd tu_repositorio
```

Haz cambios en los archivos del repositorio y agrégalos:

```
$ git add nombre_del_archivo o git add .
```

Crea un commit de tus cambios:

```
$ git commit -m "Descripción de los cambios"
```

Para enviar los commits al repositorio en GitHub, debemos empujarlos con:

```
$ git push origin nombre_de_la_rama
```

- ¿Qué es un repositorio remoto?

Un repositorio remoto es una copia de un proyecto que se encuentra en un servidor remoto, ya sea en internet o en una red. Se comparte entre varios miembros de un equipo.

Los repositorios remotos son útiles para colaborar entre usuarios en un proyecto. Permiten realizar cambios en el código fuente de una aplicación, y mantenerlos actualizados con el repositorio remoto.

Características de los repositorios remotos

- Se pueden alojar en un servidor externo, en internet o en la misma máquina en una ruta diferente.
- Sirven como copia de seguridad.
- Permiten que otros usuarios accedan a ellos y realicen cambios.
- Son versiones del proyecto que están hospedadas en internet o en cualquier otra red.

GitHub, un proveedor de repositorios remotos

GitHub es uno de los proveedores más usados para alojar repositorios Git.

- ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git, puedes usar el comando `$ git remote add` (Este comando se debe ejecutar dentro del directorio donde está el repositorio).

Paso a paso

1. Entrar al directorio donde está el repositorio
2. Ejecutar el comando `git remote add`
3. Especificar un nombre remoto, por ejemplo, origin

- ¿Cómo empujar cambios a un repositorio remoto?

Para empujar cambios a un repositorio remoto, puedes usar el comando `$ git push`

`$ git push "nombre del repositorio" "nombre de la rama"`

(Por ejemplo, “`git push origin main`” para insertar los cambios locales en el repositorio en línea)

- ¿Cómo tirar de cambios de un repositorio remoto?

Para obtener los cambios de un repositorio remoto en Git, puedes usar el comando `git pull`. Este comando descarga el contenido de un repositorio remoto y actualiza el repositorio local

`$ git pull origin nombre_de_la_rama`

(Por ejemplo, si estás trabajando en la rama main: `$ git pull origin main`)

- ¿Qué es un fork de repositorio?

Un fork de repositorio es una copia exacta de un repositorio que se crea para colaborar en un proyecto.

¿Cómo se hace un fork de repositorio?

1. Ir a la página del repositorio
2. Pulsar el botón “Fork”
3. Se obtiene una copia idéntica del repositorio original, pero con una URL diferente

¿Para qué sirve un fork de repositorio?

- Para participar en un proyecto en el que no se tienen permisos de escritura
- Para colaborar con otros desarrolladores
- Para contribuir a proyectos de código abierto
- Para experimentar con nuevas características

- ¿Cómo crear un fork de un repositorio?

Para crear un fork de un repositorio en GitHub, puedes seguir estos pasos:

Ir a la página del repositorio que quieres copiar

1. Hacer clic en el botón Fork
2. Si es necesario, nombrar y describir el fork
3. Hacer clic en Create fork
4. Un fork es una copia exacta de un repositorio que se crea en tu cuenta. Esto te permite contribuir a un proyecto sin tener permisos de escritura.

Ventajas de hacer un fork:

- Puedes trabajar en tu propia copia sin necesidad de acceder al repositorio central
- El propietario del repositorio puede gestionar fácilmente las contribuciones de varias personas

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

Para enviar una solicitud de extracción (pull request) a un repositorio en GitHub, puedes seguir estos pasos:

1. Ir a la página principal del repositorio
2. Seleccionar la rama que contiene las confirmaciones en el menú "Rama"
3. Hacer clic en "Comparar y solicitud de incorporación de cambios" en el banner amarillo
4. Seleccionar la rama base y la rama de comparación
5. Escribir un título y una descripción
6. Hacer clic en "Crear solicitud de incorporación de cambios"

Si la solicitud de extracción es aceptada, se recibirá un correo electrónico.

- ¿Cómo aceptar una solicitud de extracción?

El autor del repositorio puede visualizar en su pull request el mensaje que la persona envió, con el objetivo de aportar modificaciones a la rama original, y en el caso de estar conforme, puede aceptar el Merge del pull request, y de esta manera sumara a su repositorio original los cambios hechos por otro usuario en modo de contribución.

- ¿Qué es una etiqueta en Git?

En Git, una etiqueta es una marca que se aplica a una confirmación para identificarla con información relevante para otros usuarios del repositorio

- ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta en Git, puedes usar el comando git tag.

Paso 1

1. Especifica el nombre de la etiqueta. Por ejemplo:
\$ git tag v1.4.

Paso 2

1. Opcionalmente, puedes usar las opciones -a o -m para crear una etiqueta anotada.
2. Si usas -a, puedes usar -m para especificar el mensaje de la etiqueta. Por ejemplo:
\$ git tag -a v1.4 -m 'my version 1.4'.
3. Si no usas -a, -s ni -m, creas una etiqueta ligera. Por ejemplo:
\$ git tag v1.4-lw.

Paso 3

1. Para obtener una lista de las etiquetas almacenadas, puedes usar el comando:
\$ git tag .

Paso 4

1. Para obtener información sobre una etiqueta, puedes usar el comando:
\$ git show .

Las etiquetas son útiles para etiquetar el estado del repositorio completo, como cuando se lanza una nueva versión de un software.

- ¿Cómo enviar una etiqueta a GitHub?

Para enviar etiquetas a GitHub, puedes usar el comando `git push` con la opción: `--tags`

Paso a paso para enviar etiquetas con `git push`:

1. Crea las etiquetas localmente
2. Ejecuta el comando `$ git push origin [etiqueta]`
3. Para empujar todas las etiquetas creadas, usar: `--tags` del comando `git push`

- ¿Qué es un historial de Git?

El historial de Git es una secuencia de todos los cambios realizados en un repositorio de Git. Cada cambio en el repositorio se guarda como un commit, y cada commit contiene información sobre el estado del proyecto en un momento específico, incluyendo:

Identificador del commit

Autor

Fecha de realización

Mensaje enviado

- ¿Cómo ver el historial de Git?

Esto lo conseguimos con el comando de Git:

`$ git log`

Con tipear este comando en el bash de Git podremos apreciar el histórico de commits, estando situados en la carpeta de nuestro proyecto. El listado de commits estará invertido, es decir, los últimos realizados aparecen los primeros.

El comando `git log --oneline` es una forma compacta de visualizar el historial de commits en un repositorio Git. Muestra un resumen conciso de los commits recientes, con cada commit representado en una sola línea.

Si tu proyecto ya tiene muchos commits, quizás no quieras mostrarlos todos, ya que generalmente no querrás ver cosas tan antiguas como el origen del repositorio. Para ver un número de logs determinado introducimos ese número como opción, con el signo "-" delante (-1, -8, -12...). Por ejemplo, esto muestra los últimos tres commits: `git log -3`

Si queremos que el log también nos muestre los cambios en el código de cada commit podemos usar la opción -p. Esta opción genera una salida mucho más larga, por lo que seguramente nos tocará movernos en la salida con los cursores y usaremos CTRL + Z para salir.

```
$ git log -2 -p
```

- ¿Cómo buscar en el historial de Git?

Para buscar en el historial de commits de Git, puedes utilizar varios comandos y opciones que te permiten filtrar y localizar commits específicos.

Para buscar commits que contengan una palabra o frase específica en el mensaje de commit, usa git log con la opción -grep:

```
$ git log --grep="palabra clave"
```

Para buscar commits que han modificado un archivo específico, usa git log seguido del nombre del archivo:

```
$ git log -- nombre_del_archivo
```

Para buscar commits en un rango de fechas específico, usa las opciones --since y --until:

```
$ git log --since="2024-01-01" --until="2024-01-31"
```

Para encontrar commits hechos por un autor específico, usa --author:

```
$ git log --author="Nombre del Autor"
```

- ¿Cómo borrar el historial de Git?

El comando \$ git reset se utiliza sobre todo para deshacer las cosas, como posiblemente puedes deducir por el verbo. Se mueve alrededor del puntero HEAD y opcionalmente cambia el index o área de ensayo y también puede cambiar opcionalmente el directorio de trabajo si se utiliza - hard. Esta última opción hace posible que este comando pueda perder tu trabajo si se usa incorrectamente, por lo que asegúrese de entenderlo antes de usarlo.

Existen distintas formas de utilizarlo:

- \$ git reset -> Quita del stage todos los archivos y carpetas del proyecto.
- \$ git reset nombreArchivo > Quita del stage el archivo indicado.
- \$ git reset nombreCarpeta/ > Quita del stage todos los archivos de esa carpeta.
- \$ git reset nombreCarpeta/nombreArchivo > Quita ese archivo del stage (que a la vez está dentro de una carpeta).
- \$ git reset nombreCarpeta/*.extensión > Quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta del stage.

- ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un espacio de almacenamiento que solo es accesible para el usuario y las personas a las que se le otorgue permiso.

Características de los repositorios privados en GitHub

- Son cifrados en reposo y en vuelo.
- Se pueden crear de forma gratuita.
- Permiten tener hasta tres colaboradores.
- Se pueden invitar a colaboradores a través de un correo electrónico.
- Se pueden personalizar las opciones de visibilidad.

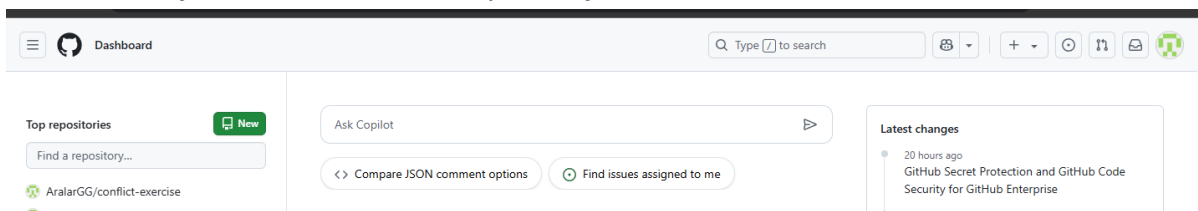
- ¿Cómo crear un repositorio privado en GitHub?

Pasos:

Inicia sesión en GitHub

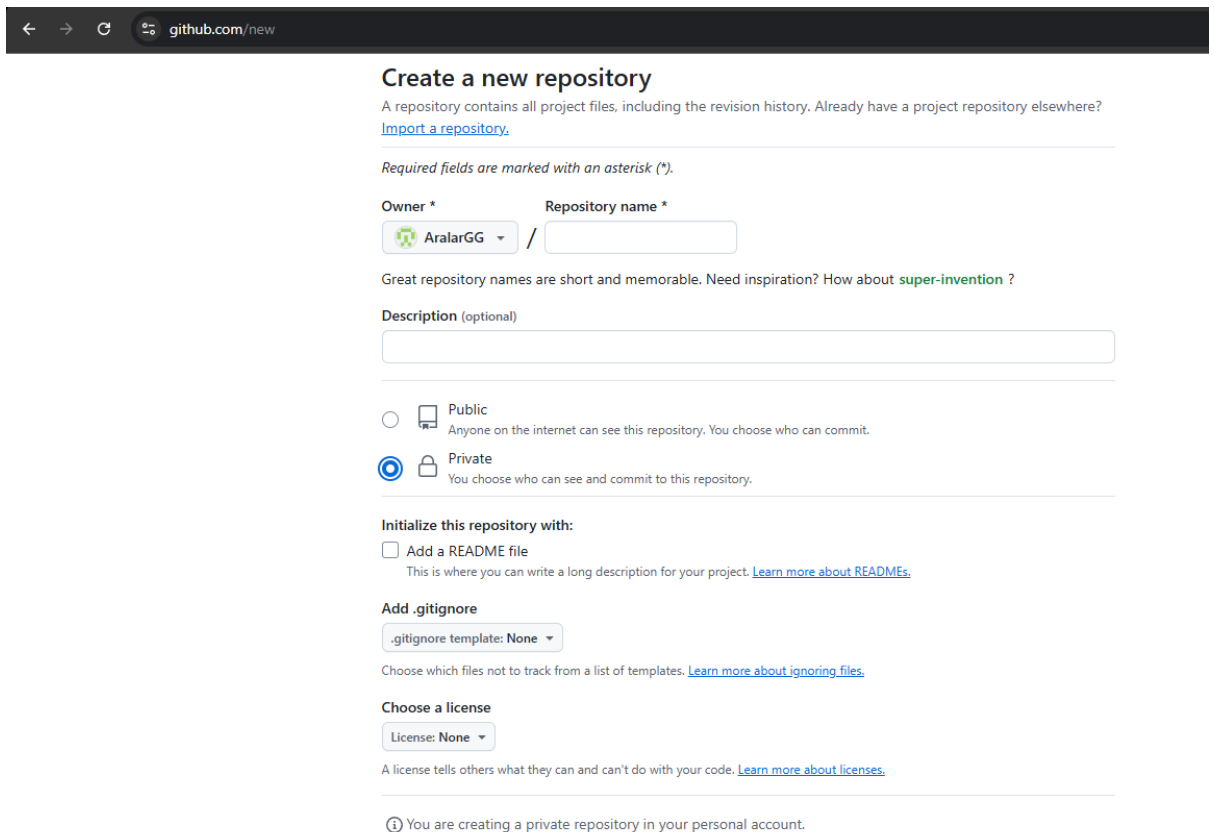
Ingresa a la página de creación de repositorios:

En la esquina superior derecha de la página principal, debes hacer clic en el botón “+” y seleccionar “New Repository” o hacer clic en “New”:



Completar la información del repositorio (nombre del repositorio, descripción)

Seleccionar la configuración de privacidad:




github.com/new

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 *

 AralarGG /

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

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

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

Choose a license

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

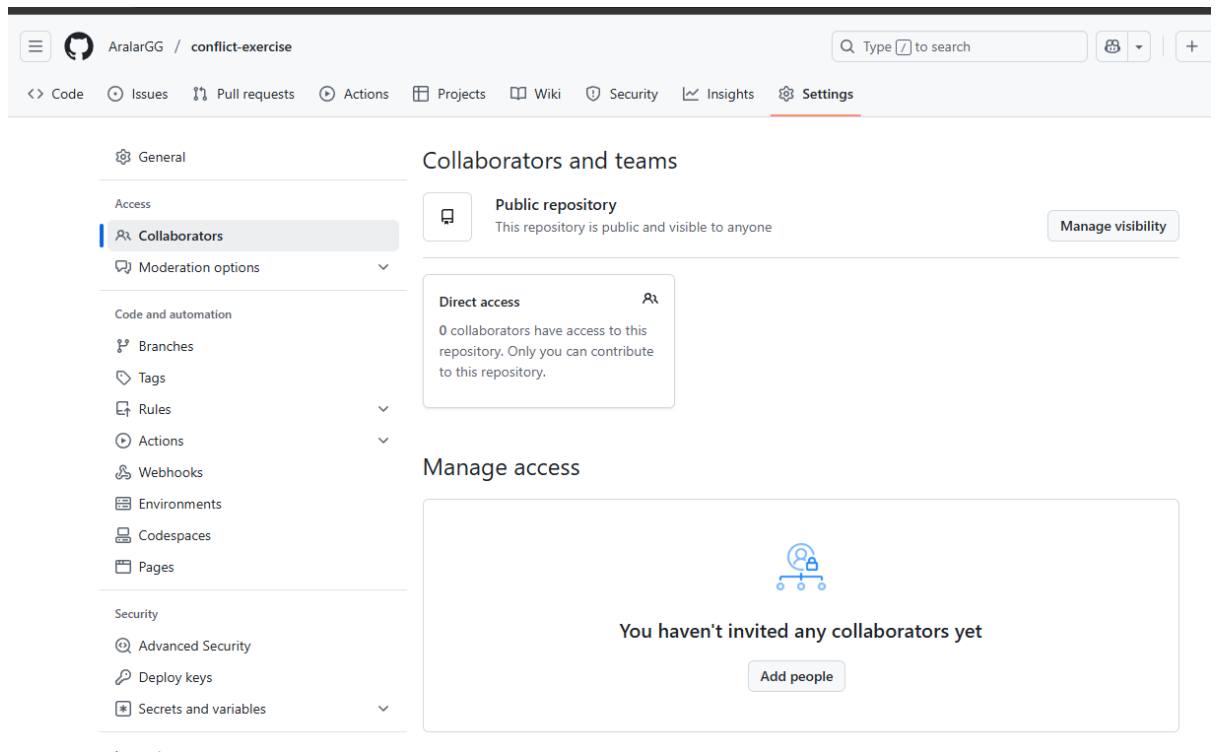
 You are creating a private repository in your personal account.

Esto asegura que el repositorio será privado y solo accesible para los colaboradores que dispongan de esta información.

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

Invitar a alguien a un repositorio privado en GitHub es un proceso sencillo, pero requiere permisos adecuados.

Accede al repositorio, haz clic en la pestaña "Settings" del repositorio. Está en la parte superior del repositorio, junto a las pestañas como "Code" y "Issues"



En la sección "Collaborators", haz clic en el botón "Add people" e ingresa el nombre de usuario de GitHub de la persona que deseas invitar. Selecciona el nivel de acceso que deseas otorgar: Read, Triage, Write, Maintain, o Admin. Haz clic en el botón "Add" para enviar la invitación.

- ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un espacio donde se puede almacenar y compartir código, archivos, y revisiones de forma que sea accesible para cualquier persona en internet

- ¿Cómo crear un repositorio público en GitHub?

De la misma manera que creamos el repositorio privado, solo que esta vez debemos escoger la opción "Public", para que cualquier persona pueda tener acceso.

New repository

Q Type / 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 *

AralarGG

/

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

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

- ¿Cómo compartir un repositorio público en GitHub?

Una vez creado el repositorio público, el cual queremos compartir solo resta copiar y pegar la URL de dicho repositorio que se encuentra en el apartado CODE como enseña la imagen a continuación:

AralarGG / UTN-TUPaD-P1

Q Type / to search

<> Code

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

UTN-TUPaD-P1

Public

forked from [sbruselario/UTN-TUPaD-P1](#)

Pin

Watch 0

main

1 Branch

0 Tags

Go to file

Add file

<> Code

This branch is up to date with [sbruselario/UTN-TUPaD-P1:main](#).

sbruselario

¡Bienvenid@ a Programación 1!

01 Estructuras Secuenciales

¡Bienvenid@ a P

02 Trabajo Colaborativo

¡Bienvenid@ a P

03 Estructuras Condicionales

¡Bienvenid@ a P

04 Estructuras Repetitivas

¡Bienvenid@ a P

05 E

P

Local

Codespaces

Clone

HTTPS SSH GitHub CLI

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

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

2) Crear un repositorio.

Dale un nombre al repositorio.


Elige 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 *	Repository name *
 AralarGG	/ <input type="text" value="Actividad2"/>
	✓ Actividad2 is available.

Great repository names are short and memorable. Need inspiration? How about [urban-rotary-phone](#) ?

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.](#)

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).

```
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo> git clone https://github.com/
AralarGG/Actividad2.git
Cloning into 'Actividad2'...
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.
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo> cd Actividad2
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad2>
echo "Este es mi archivo (Santiago)" > mi-archivo.txt
```

```

PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad> e
cho "Este es mi archivo (Santiago)" > mi-archivo.txt
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad2>g
it add .
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad2>g
it status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   mi-archivo.txt

PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad> g
it commit -m "Agregue mi-archivo.txt"
[main 3c7f314] Agregue mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt

```

.. Programación I > Primer Repo-git > UTN-TUPaD-P1 > 02 Trabajo Colaborativo > Actividad2 >

Nombre	Fecha de modificación	Tipo	Tamaño
.git	31/3/2025 01:54	Carpeta de archivos	
mi-archivo	31/3/2025 01:52	Documento de tex...	1 KB
README	31/3/2025 01:49	Archivo de origen ...	1 KB

Actividad2 Public

[Pin](#) [Unwatch](#) 1

main 1 Branch 0 Tags

Go to file

Add file

Code

AralarGG Initial commit

ed538a7 · 10 minutes ago

1 Commit

README.md

Initial commit

10 minutes ago

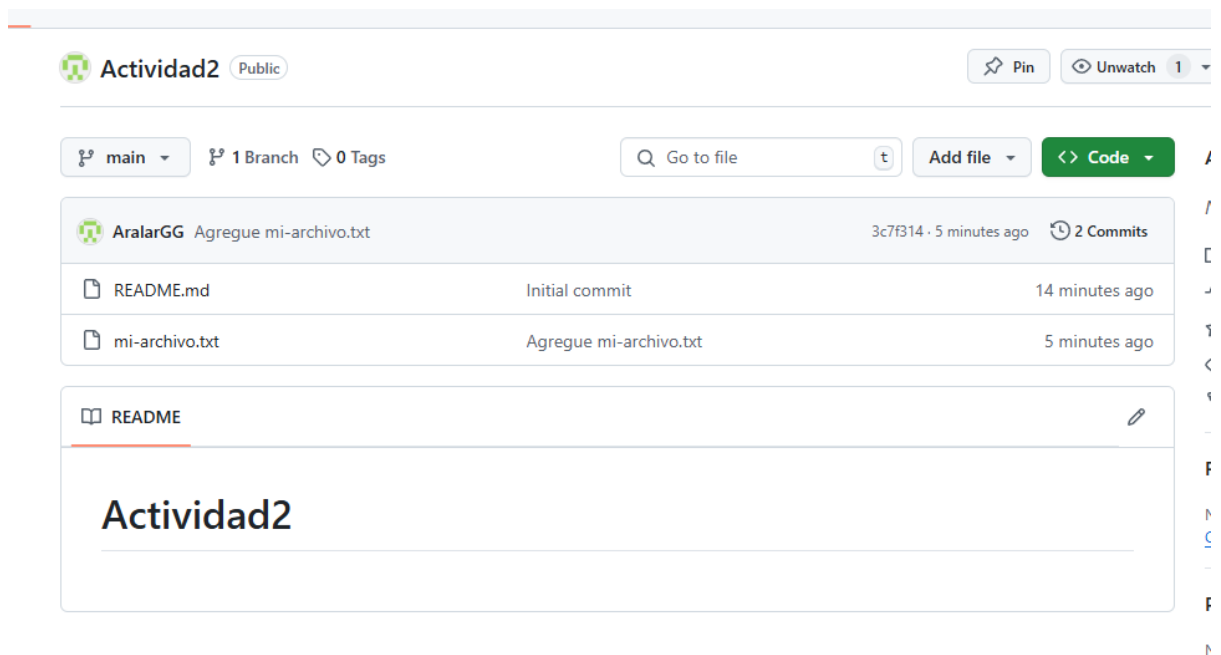
README

Actividad2

```

PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad2>g
it push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 349 bytes | 349.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/AralarGG/Actividad2.git
ed538a7..3c7f314 main -> main

```



Creando Branchs

Crear una Branch

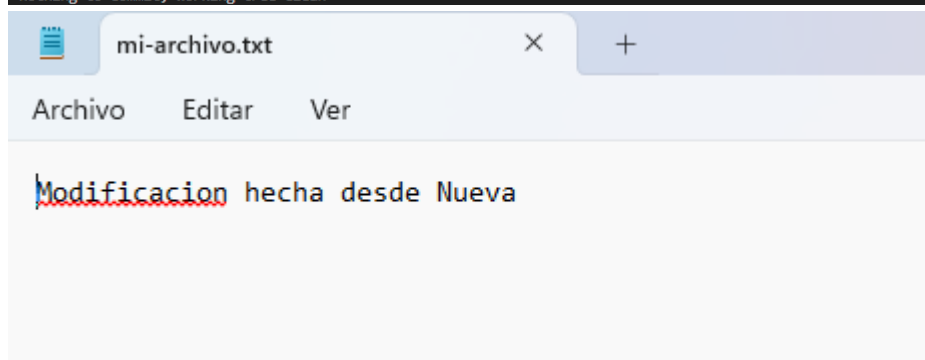
Realizar cambios o agregar un archivo

Subir la Branch

```
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad2> git checkout -b Nueva
Switched to a new branch 'Nueva'
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad2> git branch
* Nueva
main

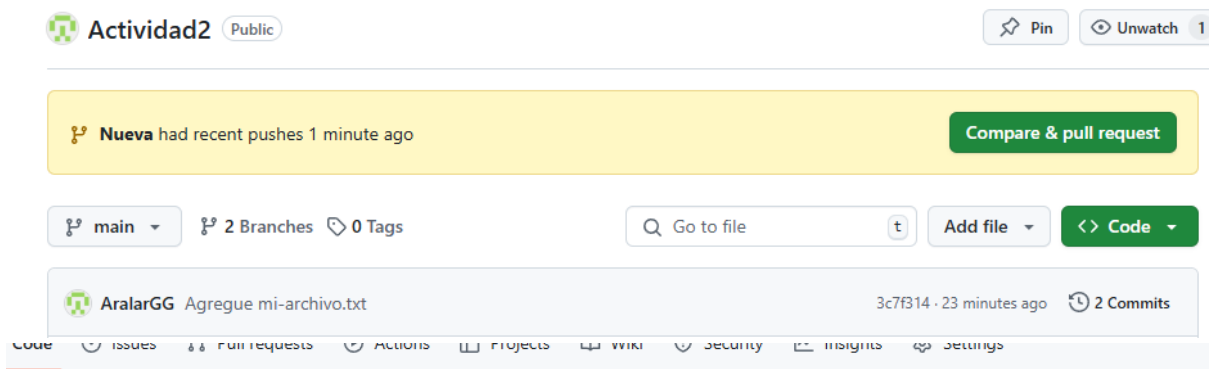
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad2> echo "Modificación hecha desde Nueva" > mi-archivo.txt
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad2> git add .
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad2> git status
On branch Nueva
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   mi-archivo.txt

PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad2> git commit -m "Modificación agregada desde Nueva"
[Nueva d590b33] Modificación agregada desde Nueva
 1 file changed, 0 insertions(+), 0 deletions(-)
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad2> git status
On branch Nueva
nothing to commit, working tree clean
```



Subimos la nueva rama a GitHub

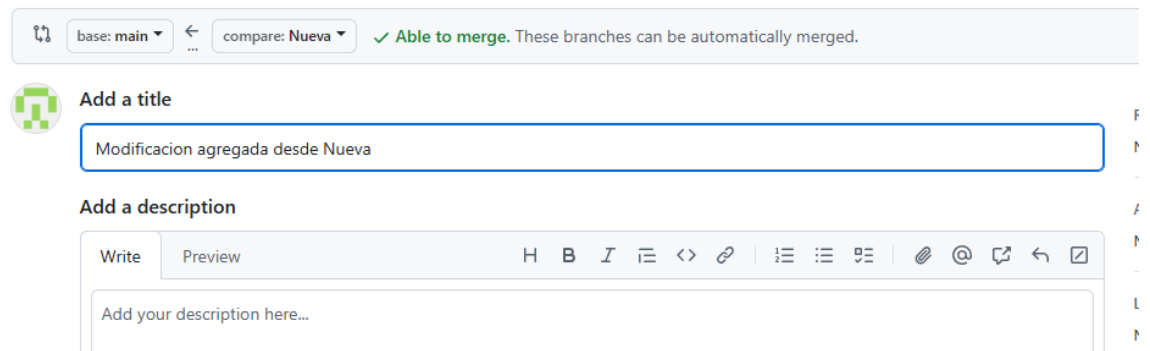

```
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Actividad2>git push origin Nueva
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), 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 'Nueva' on GitHub by visiting:
remote:   https://github.com/AralarGG/Actividad2/pull/new/Nueva
remote:
To https://github.com/AralarGG/Actividad2.git
 * [new branch]      Nueva -> Nueva
```



The screenshot shows the GitHub interface for a repository named 'Actividad2', which is public. At the top, there's a yellow notification bar stating 'Nueva had recent pushes 1 minute ago' with a 'Compare & pull request' button. Below this, the repository's main branch is 'main', with 2 branches and 0 tags. A search bar and buttons for 'Add file' and 'Code' are visible. A commit by 'AralarGG' titled 'Agregue mi-archivo.txt' is shown, dated 23 minutes ago with 2 commits. The repository navigation bar at the bottom includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#). [Learn more about diff comp](#)

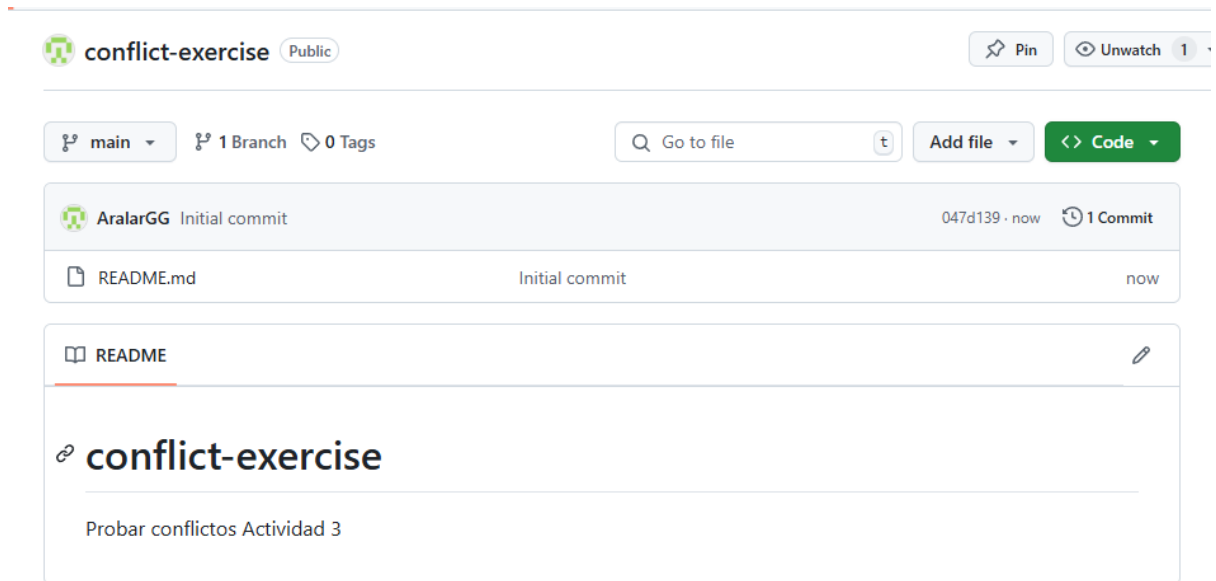


The screenshot shows the 'Open a pull request' form. At the top, it shows 'base: main' and 'compare: Nueva' with a green checkmark indicating 'Able to merge'. Below this, there's a section to 'Add a title' with a text input field containing 'Modificacion agregada desde Nueva'. Underneath is the 'Add a description' section, which includes a 'Write' tab, a 'Preview' tab, and a rich text editor with various formatting options (bold, italic, link, etc.) and a text area with the placeholder 'Add your description here...'.

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".



Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio (usualmente algo como <https://github.com/tuusuario/conflict-exercise.git>).
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando: `git clone https://github.com/tuusuario/conflict-exercise.git`

```
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
> git clone https://github.com/AralarGG/conflict-exercise.git
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.
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
> cd conflict-exercise
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
\conflict-exercise>
```

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:

```
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
\conflict-exercise> 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, por ejemplo:

Este es un cambio en la feature branch.

```
conflict-exercise > i README.md > # conflict-exercise
1 # conflict-exercise
2 Probar conflictos Actividad 3
3 Este es un cambio en la feature branch.
```

- Guarda los cambios y haz un commit:

git add README.md

git commit -m "Added a line in feature-branch"

```
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
\conflict-exercise> git add README.md
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
\conflict-exercise> git commit -m "Added a line in feature-branch"
[feature-branch 782eb95] Added a line in feature-branch
1 file changed, 1 insertion(+)
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
\conflict-exercise> |
```

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

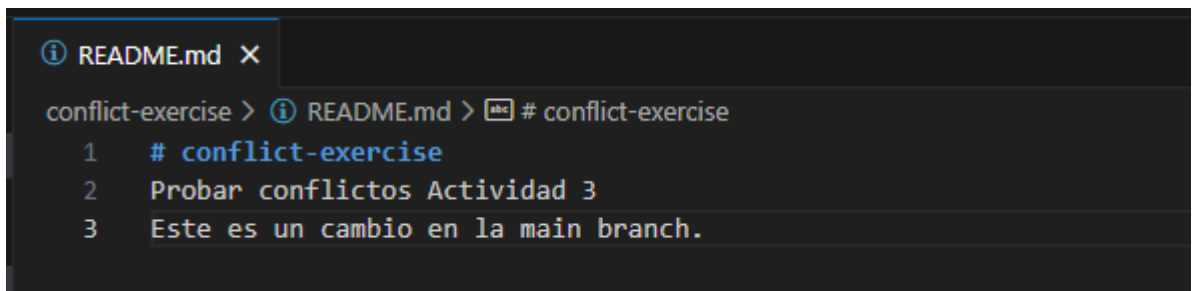
- Cambia de vuelta a la rama principal (main):

git checkout main

```
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
\conflict-exercise> 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:

Este es un cambio en la main branch.



```
conflict-exercise > i README.md > # conflict-exercise
1 # conflict-exercise
2 Probar conflictos Actividad 3
3 Este es un cambio en la main branch.
```

- Guarda los cambios y haz un commit:

git add README.md

git commit -m "Added a line in main branch"

```
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
\conflict-exercise> git add README.md
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
\conflict-exercise> git commit -m "Added a line in main branch"
[main cb71cad] Added a line in main branch
1 file changed, 1 insertion(+)
```

Paso 5: Hacer un merge y generar un conflicto

- Intenta hacer un merge de la feature-branch en la rama main:

git merge feature-branch

```
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
\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.
```

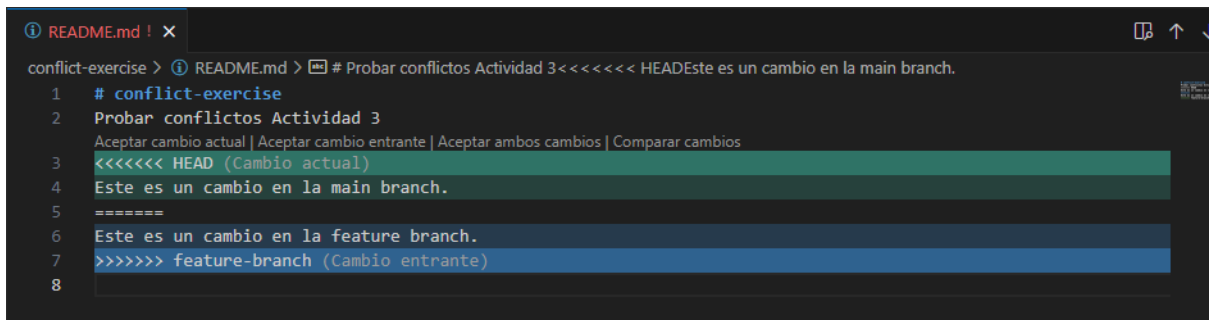
• Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md.

Paso 6: Resolver el conflicto

- Abre el archivo README.md en tu editor de texto. Verás algo similar a esto: Copiar código<<<<<< HEAD

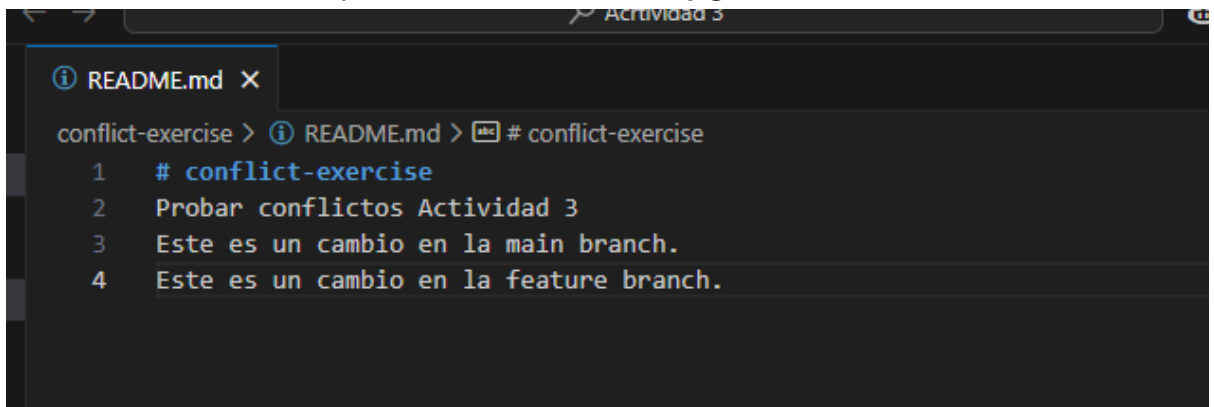
Este es un cambio en la main branch.
=====

Este es un cambio en la feature branch.
>>>>>> feature-branch

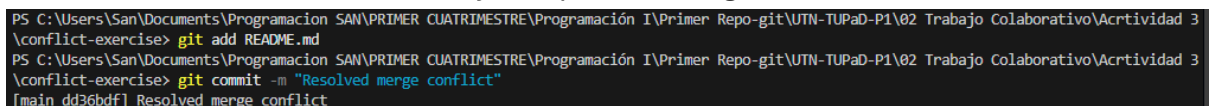


- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.

- Edita el archivo para resolver el conflicto y guarda los cambios.



- Añade el archivo resuelto y completa el merge:



Paso 7: Subir los cambios a GitHub

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

git push origin main

```
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
\conflict-exercise> git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 825 bytes | 275.00 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/AralarGG/conflict-exercise.git
047d139..dd36bdf main -> main
```

- También sube la feature-branch si deseas:

git push origin feature-branch

```
PS C:\Users\San\Documents\Programacion SAN\PRIMER CUATRIMESTRE\Programación I\Primer Repo-git\UTN-TUPaD-P1\02 Trabajo Colaborativo\Acrtividad 3
\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/AralarGG/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/AralarGG/conflict-exercise.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.

The screenshot shows the GitHub repository page for 'conflict-exercise'. At the top, there's a navigation bar with links to Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below this, the repository name 'conflict-exercise' is displayed with a 'Public' badge. On the right, there are buttons for 'Pin' and 'Unwatch'. The main content area shows the 'main' branch selected, with '2 Branches' and '0 Tags'. A search bar and 'Go to file' button are present. Below this, a commit by 'AralarGG' is shown, titled 'Resolved merge conflict', with a commit hash 'dd36bdf' and a timestamp '7 minutes ago'. The commit message is 'Resolved merge conflict'. Below the commit, the 'README' file is shown, with the title 'conflict-exercise' and the content 'Probar conflictos Actividad 3 Este es un cambio en la main branch. Este es un cambio en la feature branch.'