

- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada

1. ¿Qué es GitHub?

GitHub es una plataforma basada en la nube que permite alojar repositorios de Git, facilitando la colaboración en proyectos de desarrollo de software. Permite el control de versiones, seguimiento de cambios y trabajo en equipo.

2. ¿Cómo crear un repositorio en GitHub?

1. Inicia sesión en [GitHub](https://github.com).
2. Haz clic en el ícono "+" en la esquina superior derecha y selecciona **"New repository"**.
3. Ingresa un nombre para el repositorio y selecciona si será público o privado.
4. Opcionalmente, puedes agregar un README, .gitignore y una licencia.
5. Haz clic en **"Create repository"**.

3. ¿Cómo crear una rama en Git?

En Git, la rama "master" (o "main" en versiones más recientes) no tiene ninguna propiedad especial; es simplemente una rama más dentro del repositorio. Sin embargo, suele ser la que se genera por defecto cuando se ejecuta el comando git init.

Si deseas crear una nueva rama en tu repositorio, puedes hacerlo con el siguiente comando:

[git branch nombreDeLaRama](#)

Es importante notar que, al crear una nueva rama de esta manera, Git no cambia automáticamente a ella; seguirás en la rama en la que estabas antes de ejecutar el comando.

#### 4. ¿Cómo cambiar a una rama en Git?

Para cambiar entre ramas en un repositorio de Git, se utiliza el siguiente comando:

[git checkout nombreDeLaRama](#)

Este comando actualiza el puntero HEAD, moviéndolo a la rama especificada.

Si necesitas crear una nueva rama y cambiar a ella en un solo paso, puedes hacerlo con el siguiente comando:

[git checkout -b nombreDeLaRama](#)

#### 5. ¿Cómo fusionar ramas en Git?

Cuando trabajas con varias ramas en un repositorio, es común que necesites combinar los cambios de una en otra. Para esto, Git permite realizar fusiones entre ramas con el comando merge.

Primero, asegúrate de estar en la rama donde deseas incorporar los cambios. Si, por ejemplo, quieres fusionar la rama nuevaRama en master, primero cambia a master:

[git checkout master](#)

Luego, usa el siguiente comando para fusionar los cambios de nuevaRama en master:

[git merge nuevaRama](#)

#### 6. ¿Cómo enviar un commit a GitHub?

Un **commit** en Git representa un punto en el tiempo dentro del historial del proyecto. Este proceso permite guardar los cambios realizados en los archivos y asegurarse de que queden registrados en el repositorio.

##### **Pasos para hacer un commit:**

1. Realiza las modificaciones necesarias en los archivos del proyecto.
2. Agrega los cambios al área de preparación con el comando git add.  
Puedes incluir un archivo específico o todos los archivos modificados:

[git add nombreDelArchivo](#)

[git add .](#)

3. Una vez que los cambios estén en el área de preparación, confirma el commit con el siguiente comando:

[git commit -m "Descripción de los cambios realizados"](#)

Este proceso guarda el estado actual del código en el historial del repositorio junto con un mensaje que describe los cambios realizados.

#### 7. ¿Qué es un repositorio remoto?

Para subir un commit a un repositorio en GitHub, sigue estos pasos:

1. Clona el repositorio de GitHub en tu máquina local si aún no lo has hecho:

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

[cd tu\\_repositorio](#)

2. Realiza cambios en los archivos y agrégalos al área de preparación:

[git add nombre\\_del\\_archivo](#)

[git add .](#)

3. Crea un commit con los cambios realizados:

[git commit -m "Descripción de los cambios"](#)

4. Envía los cambios al repositorio remoto en GitHub con el siguiente comando:

[git push origin nombre\\_de\\_la\\_rama](#)

#### 8. ¿Cómo agregar un repositorio remoto a Git?

Un **repositorio remoto** es una versión del proyecto almacenada en una red o en Internet, generalmente en plataformas como GitHub, GitLab o Bitbucket. Estos repositorios permiten a múltiples desarrolladores colaborar en un mismo proyecto al compartir código y sincronizar cambios.

Trabajar con repositorios remotos implica:

- Agregar o eliminar repositorios remotos.
- Gestionar diferentes ramas remotas.
- Sincronizar cambios entre el repositorio local y el remoto.

#### 9. ¿Cómo empujar cambios a un repositorio remoto?

Para conectar un repositorio local con un remoto, usa el siguiente comando:

[git remote add nombreRemoto URL\\_del\\_Repositorio](#)

Asignar un nombre al repositorio remoto facilita su uso en futuros comandos sin necesidad de escribir la URL completa.

#### 10. ¿Cómo tirar de cambios de un repositorio remoto?

Antes de subir cambios a un repositorio remoto, es recomendable obtener la versión más reciente del código para evitar conflictos:

[git pull origin nombre\\_de\\_la\\_rama](#)

Luego, para enviar tus cambios al repositorio remoto, usa:

[git push origin nombre\\_de\\_la\\_rama](#)

Este comando sube los commits de la rama local al servidor remoto.

#### 11. ¿Qué es un fork de repositorio?

recientes de un repositorio remoto con tu rama local, utiliza:

[git pull origin nombre\\_de\\_la\\_rama](#)

Por ejemplo, si estás trabajando en la rama principal (main):

[git pull origin main](#)

Este comando sincroniza el código de tu máquina con la versión más actualizada del repositorio remoto.

#### 12. ¿Cómo crear un fork de un repositorio?

Un **fork** es una copia de un repositorio alojado en GitHub. Esta funcionalidad permite a los usuarios crear su propia versión de un proyecto sin afectar el original.

El proceso de forking es útil cuando quieres:

- Probar cambios sin modificar el código fuente original.
- Colaborar en proyectos de código abierto.
- Personalizar un repositorio para adaptarlo a tus necesidades.

Para hacer un fork de un repositorio en GitHub:

1. Encuentra el repositorio de interés.
2. Haz clic en el botón "**Fork**" en la parte superior derecha de la página.
3. GitHub creará una copia del repositorio en tu cuenta personal.

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

Si quieres practicar el proceso de fork, puedes crear una cuenta adicional en GitHub y seguir estos pasos:

1. Desde la cuenta secundaria, haz un **fork** de uno de los repositorios de tu cuenta principal.

2. Clona el fork en tu computadora:

```
git clone https://github.com/tu_usuario/nombre_del_fork.git
```

3. Realiza modificaciones en los archivos y agrégalos al área de preparación:

[git add nombre\\_del\\_archivo](#)

4. Confirma los cambios con un commit:

```
git commit -m "Descripción de las mejoras realizadas"
```

5. Sube los cambios al fork con:

[git push origin master](#)

14. ¿Cómo aceptar una solicitud de extracción?

Para proponer cambios en un repositorio en GitHub, se utiliza una **solicitud de extracción** (*pull request*). Esto permite que el propietario del repositorio revise y, si lo considera conveniente, fusione los cambios en el proyecto principal.

**Pasos para crear un *pull request*:**

1. Accede a la pestaña **Pull requests** en el repositorio de GitHub.
2. Haz clic en **New pull request**.
3. Se abrirá una vista previa en la que podrás comparar los cambios de tu rama con la versión original.
4. Presiona **Create pull request**.
5. Escribe un título y una descripción explicando los cambios y por qué deberían añadirse al proyecto.
6. Finalmente, haz clic en **Submit pull request** para enviarlo al propietario del repositorio.

15. ¿Qué es un etiqueta en Git?

Git permite asignar **etiquetas** (*tags*) a versiones específicas del historial del repositorio. Esto es útil para marcar hitos importantes, como versiones de lanzamiento (ejemplo: v1.0).

Existen dos tipos principales de etiquetas en Git:

- **Etiquetas ligeras:** Son simplemente referencias a un commit específico.
- **Etiquetas anotadas:** Almacenan información adicional, como el nombre del creador, la fecha y un mensaje descriptivo.

16. ¿Cómo crear una etiqueta en Git?

Para añadir una etiqueta en Git, puedes elegir entre una **ligera** o una **anotada**.

**Crear una etiqueta ligera:**

[git tag v1.0](#)

**Crear una etiqueta anotada:**

[git tag -a v1.0 -m "Versión 1.0 del proyecto"](#)

Las etiquetas anotadas son más recomendadas, ya que almacenan información adicional útil sobre el commit.

17. ¿Cómo enviar una etiqueta a GitHub?

Después de crear una etiqueta en tu repositorio local, debes enviarla al repositorio remoto para que esté disponible en GitHub.

Para subir una etiqueta específica:

[git push origin v1.0](#)

18. ¿Qué es un historial de Git?

El **historial de Git** es el registro de todos los cambios realizados en un repositorio. Cada modificación se guarda como un **commit**, que almacena información detallada sobre el estado del proyecto en un momento dado.

Cada commit en el historial contiene:

- Un identificador único (*hash*).
- El nombre del autor.
- La fecha en que se realizó el cambio.
- Un mensaje descriptivo sobre la modificación.

Para visualizar el historial de un repositorio, usa el siguiente comando:

[git log](#)

#### 19. ¿Cómo ver el historial de Git?

Para revisar el historial de cambios en un repositorio Git, puedes utilizar el siguiente comando:

[git log](#)

Este comando muestra todos los *commits* realizados en el proyecto, en orden cronológico inverso (los más recientes aparecen primero).

Si prefieres una versión más resumida, puedes usar:

[git log --oneline](#)

Esto muestra cada commit en una sola línea, incluyendo su identificador y mensaje.

Si deseas limitar la cantidad de *commits* mostrados, puedes indicar el número de registros que quieres ver:

[git log -3](#)

#### 20. ¿Cómo buscar en el historial de Git?

Git permite filtrar el historial de *commits* según diferentes criterios:

- **Buscar por palabra clave en el mensaje de commit:**

[git log --grep="palabra clave"](#)

- **Buscar cambios en un archivo específico:**

[git log -- nombre del archivo](#)

#### 21 ¿Cómo borrar el historial de Git?

Si necesitas deshacer cambios en Git, puedes usar el comando `git reset`, que permite mover el puntero HEAD y modificar el estado del área de preparación (*staging area*).

Algunas formas de utilizar `git reset` son:

- **Quitar todos los archivos del área de preparación:**

`git reset`

- **Quitar un archivo específico del *stage*:**

[git reset nombreArchivo](#)

- **Quitar todos los archivos de una carpeta del *stage*:**

[git reset nombreCarpeta/](#)

22. ¿Qué es un repositorio privado en GitHub?

Un **repositorio privado** en GitHub es un espacio donde el código solo está disponible para usuarios autorizados. A diferencia de los repositorios públicos, estos no pueden ser vistos ni clonados por otras personas a menos que se les otorguen permisos explícitos.

Son ideales para proyectos que contienen información confidencial o que aún no están listos para ser compartidos públicamente.

23. ¿Cómo crear un repositorio privado en GitHub?

1. Inicia sesión en GitHub.
2. Haz clic en el botón + en la parte superior derecha y selecciona **New repository**.
3. Introduce un nombre para el repositorio y una descripción opcional.
4. Selecciona la opción **Private**.
5. Configura otras opciones si lo deseas (como añadir un README o una licencia).
6. Haz clic en **Create repository**.

Ahora, solo los colaboradores que invites podrán acceder al repositorio.

24. ¿Cómo invitar a alguien a un repositorio privado en GitHub?

Para compartir un repositorio privado con otro usuario, sigue estos pasos:

1. Ve al repositorio en GitHub.
2. Dirígete a la pestaña **Settings**.
3. En la barra lateral, selecciona **Manage access**.
4. Haz clic en **Invite a collaborator**.
5. Escribe el nombre de usuario o correo electrónico de la persona a la que deseas invitar.
6. Presiona **Add collaborator** y espera a que la persona acepte la invitación.



25. ¿Qué es un repositorio público en GitHub?

Un **repositorio público** en GitHub es un proyecto cuyo código puede ser visto, clonado y utilizado por cualquier persona en Internet.

Estos repositorios son útiles para compartir código abierto, colaborar con otros desarrolladores y permitir que cualquier persona contribuya con mejoras o sugerencias.

26. ¿Cómo crear un repositorio público en GitHub?

Para crear un repositorio público, sigue estos pasos:

1. **Inicia sesión en GitHub.**
2. **Accede a la creación de repositorios:** En la esquina superior derecha, haz clic en + y selecciona **New repository**.
3. **Define el repositorio:**
  - Introduce un nombre para tu proyecto.
  - Agrega una descripción opcional.
  - Selecciona **Public**.
4. **Opcionalmente, inicializa el repositorio con:**
  - Un archivo README (para describir el proyecto).
  - Una licencia (si deseas compartirlo bajo ciertas condiciones).
  - Un archivo .gitignore (para excluir archivos innecesarios).
5. **Haz clic en "Create repository"**

27. ¿Cómo compartir un repositorio público en GitHub?

**Abre GitHub** y ve a la página del repositorio que deseas compartir.

**Copia la URL** desde la barra de direcciones del navegador.

- El formato de la URL es:

Copiar código

[https://github.com/tu\\_usuario/nombre\\_del\\_repositorio](https://github.com/tu_usuario/nombre_del_repositorio)

**Comparte el enlace** con otras personas a través de correo, redes sociales o mensajería.

2) <https://github.com/CamilaRL1/Actividad-N-2>



```
MINGW64:/c/Pseint 2025/Git/Unidad2

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (master)
$ git init
Reinitialized existing Git repository in C:/Pseint 2025/Git/Unidad2/.git/

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (master)
$ git add .

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (master)
$ git commit -m "Creada la actividad"
[master 49c37a7] Creada la actividad
1 file changed, 9 insertions(+), 1 deletion(-)

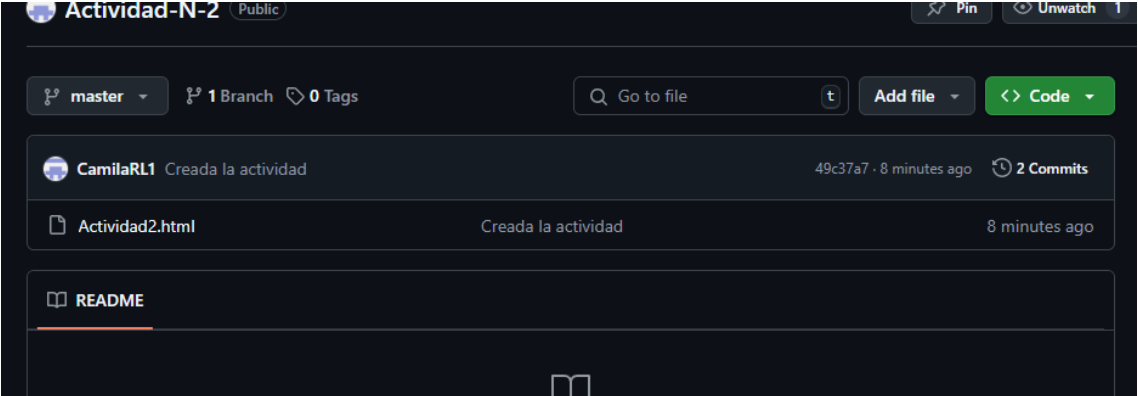
camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (master)
$
```

## Almacenado en GitHub

```
camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (master)
$ git remote add origin https://github.com/CamilaRL1/Actividad-N-2.git

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (master)
$ git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 526 bytes | 526.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/CamilaRL1/Actividad-N-2.git
 * [new branch]      master -> master

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (master)
$
```



## Crear una Branch

```

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (master)
$ git branch
* master

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (master)
$ git switch -c "Branc1"
Switched to a new branch 'Branc1'

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branc1)
$ git add .

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branc1)
$ git commit -m "Se creo una nueva modificacion en la rama Branc1"
bash: git: command not found

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branc1)
$ git commit -m "Se creo una nueva modificacion en la rama Branc1"
[Branc1 00c20e0] Se creo una nueva modificacion en la rama Branc1
1 file changed, 1 insertion(+), 1 deletion(-)

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branc1)
$ git add .

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branc1)
$ git log
commit 00c20e084332ea3177bf44d02a89cb7c04374292 (HEAD -> Branc1)
Author: Camila <rissicamila07@gmail.com>
Date: Tue Apr 1 23:24:03 2025 -0300

    Se creo una nueva modificacion en la rama Branc1

commit 49c37a7079e13d3b34b3d4bce45ef97e9a16ca11 (origin/master, master)
Author: Camila <rissicamila07@gmail.com>
Date: Tue Apr 1 23:05:08 2025 -0300

    Creada la actividad

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 ((00c20e0...))
$ git log
commit 00c20e084332ea3177bf44d02a89cb7c04374292 (HEAD, Branc1)
Author: Camila <rissicamila07@gmail.com>
Date: Tue Apr 1 23:24:03 2025 -0300

    Se creo una nueva modificacion en la rama Branc1

commit 49c37a7079e13d3b34b3d4bce45ef97e9a16ca11 (origin/master, master)
Author: Camila <rissicamila07@gmail.com>
Date: Tue Apr 1 23:05:08 2025 -0300

    Creada la actividad

commit 2cd1b7b8873dc991105c508ca4bab3ffa0f36dbe
Author: Camila <rissicamila07@gmail.com>
Date: Tue Apr 1 22:57:45 2025 -0300

    AgregadoActividad2.html

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 ((00c20e0...))
$ git branch
* (HEAD detached at 00c20e0)
  Branc1
  master

```

```

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 ((41bb055...))
$ git branch
  Branc1
* Branch-rama
  master

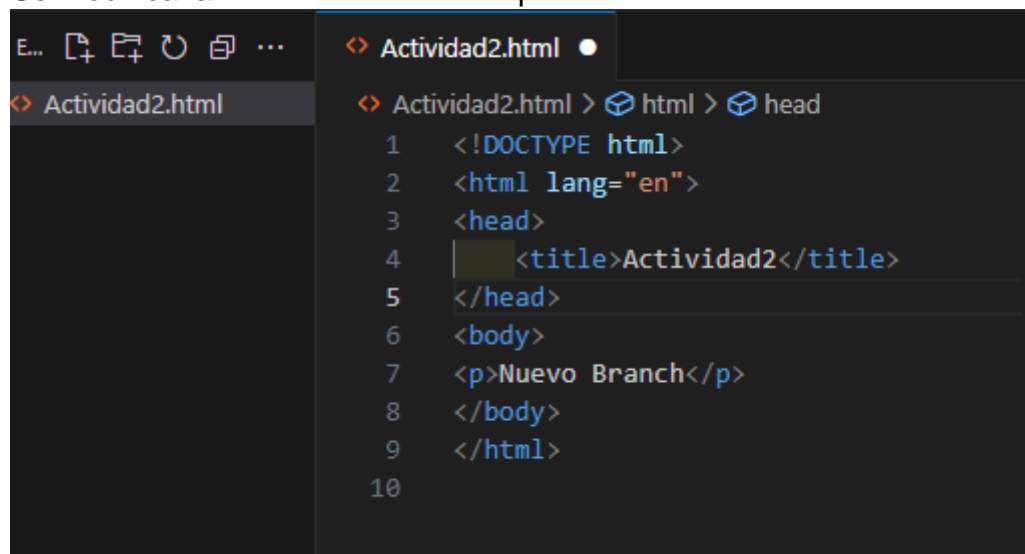
camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branch-rama)
$ git add .

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branch-rama)
$ git commit -m "Se modifiko la branch"
[Branch-rama d8877c2] Se modifiko la branch
1 file changed, 2 insertions(+), 2 deletions(-)

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branch-rama)
$ git add .S

```

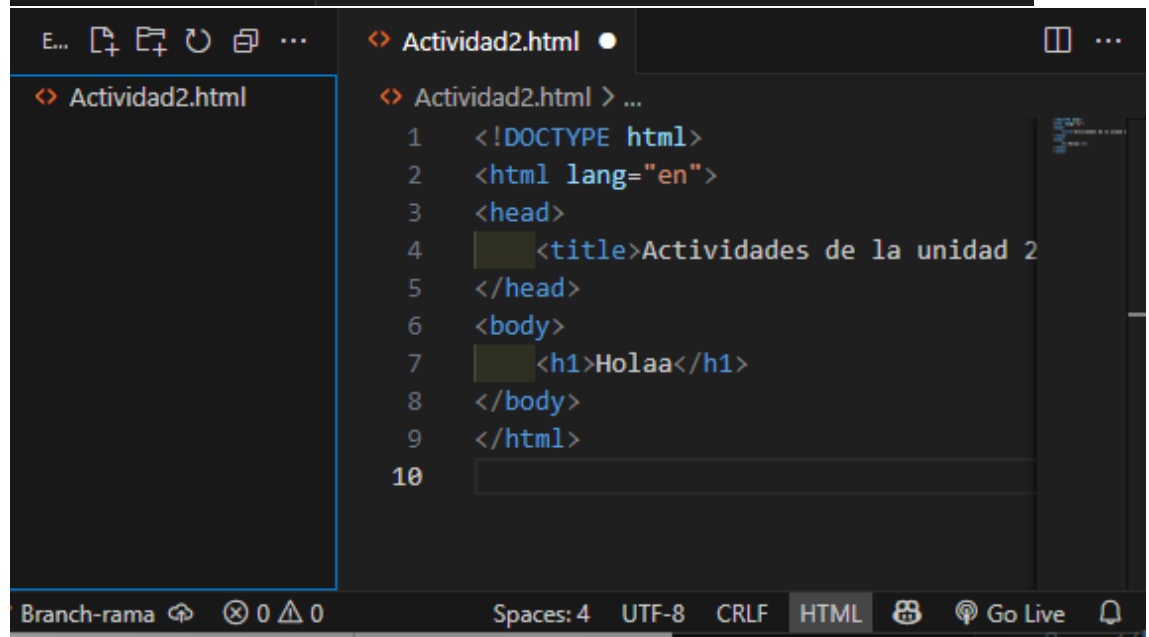
Se modifica la rama "Branch-rama" por "Branc1"



```

<?xml version="1.0" encoding="UTF-8" ?>
<html>
  <head>
    <title>Actividad2</title>
  </head>
  <body>
    <p>Nuevo Branch</p>
  </body>
</html>

```



```

<?xml version="1.0" encoding="UTF-8" ?>
<html lang="en">
  <head>
    <title>Actividades de la unidad 2</title>
  </head>
  <body>
    <h1>Holaa</h1>
  </body>
</html>

```

3)Git Hub enlace: <https://github.com/CamilaRL1/-conflict-exercise>

Paso2

```

MINGW64:/c/Pseint 2025/Git/Unidad2

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branch-rama)
$ git clone https://github.com/CamilaRL1/-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.

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branch-rama)
$ cd conflict-exercise
bash: cd: conflict-exercise: No such file or directory

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branch-rama)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (feature-branch)
$ s

```

### Paso 3

```

MINGW64:/c/Pseint 2025/Git/Unidad2

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.

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branch-rama)
$ cd conflict-exercise
bash: cd: conflict-exercise: No such file or directory

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branch-rama)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (feature-branch)
$ git add README.md

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (feature-branch)
$ git commit -m "Se a adio una nueva linea en la rama de funciones "
[feature-branch 1f6b08a] Se a adio una nueva linea en la rama de funciones
1 file changed, 2 insertions(+)
create mode 100644 README.md

camil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (feature-branch)
$

```

### Paso 4

```

MINGW64:/c/Pseint 2025/Git/Unidad2
int: If you added this path by mistake, you can remove it from the
int: index with:
int:
int:   git rm --cached -conflict-exercise
int:
int: See "git help submodule" for more information.
int: Disable this message with "git config set advice.addEmbeddedRepo false"

amil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (Branch-rama)
git checkout -b main
Switched to a new branch 'main'

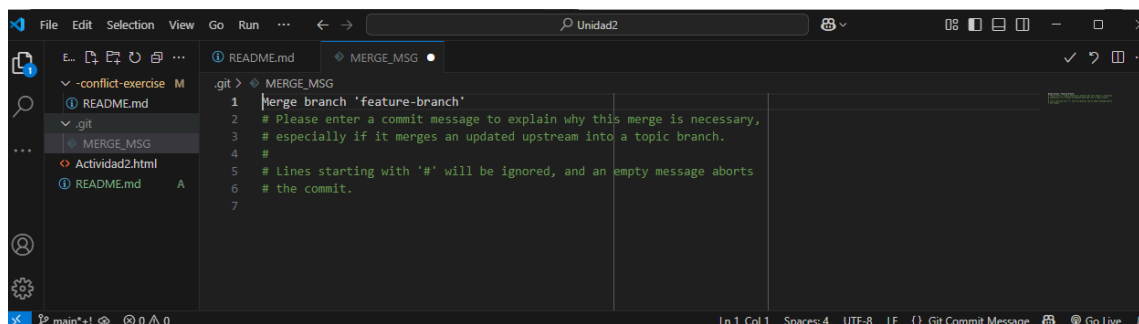
amil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (main)
git add README.md
fatal: pathspec 'README.md' did not match any files

amil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (main)
git commit -m "Se a adio una nueva linea en la rama main branch"
main 1300575] Se a adio una nueva linea en la rama main branch
1 file changed, 1 insertion(+)
create mode 160000 -conflict-exercise

amil@LAPTOP-MFCNQQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (main)

```

## Paso 5



```

.git > MERGE_MSG
1 Merge branch 'feature-branch'
2 # Please enter a commit message to explain why this merge is necessary,
3 # especially if it merges an updated upstream into a topic branch.
4 #
5 # Lines starting with '#' will be ignored, and an empty message aborts
6 # the commit.
7

```

## Paso 6

```

MINGW64:/c/Pseint 2025/Git/Unidad2
$ git add README.md
fatal: pathspec 'README.md' did not match any files

camil@LAPTOP-MFCNQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (main)
$ git commit -m "Se a adio una nueva linea en la rama main branch"
[main 1300575] Se a adio una nueva linea en la rama main branch
1 file changed, 1 insertion(+)
create mode 160000 -conflict-exercise

camil@LAPTOP-MFCNQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (main)
$ git merge feature-branch
Merge made by the 'ort' strategy.
 README.md | 2 ++
1 file changed, 2 insertions(+)
create mode 100644 README.md

camil@LAPTOP-MFCNQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (main)
$

camil@LAPTOP-MFCNQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (main)
$

camil@LAPTOP-MFCNQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (main)
$ git add .

```

## Paso 7

```

MINGW64:/c/Pseint 2025/Git/Unidad2

no changes added to commit (use "git add" and/or "git commit -a")

camil@LAPTOP-MFCNQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (main)
$ git add .

camil@LAPTOP-MFCNQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (main)
$ git push origin main
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 4 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (16/16), 1.62 KiB | 829.00 KiB/s, done.
Total 16 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
remote:
remote: Create a pull request for 'main' on GitHub by visiting:
remote:   https://github.com/CamilaRL1/Actividad-N-2/pull/new/main
remote:
To https://github.com/CamilaRL1/Actividad-N-2.git
 * [new branch]      main -> main

camil@LAPTOP-MFCNQ72 MINGW64 /c/Pseint 2025/Git/Unidad2 (main)
$
200 "message": " usuario registrado exitosamente!"

```