

Programacion 1

Trabajo Practico N°2: Git y GitHub

Actividades

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

- ¿Qué es GitHub?

GitHub es una plataforma en línea que permite a desarrolladores y equipos trabajar juntos en proyectos de software utilizando Git, un sistema de control de versiones. En GitHub, puedes almacenar tu código, colaborar con otros programadores, gestionar problemas (issues), hacer revisiones de código y desplegar proyectos. Es ampliamente utilizado tanto en el mundo profesional como en la educación.

- ¿Cómo crear un repositorio en GitHub?

Un repositorio es donde almacenas tu código y el historial de cambios. Para crearlo:

- 1) Inicia sesión en GitHub.
- 2) Ve a "New" o dirígete a <https://github.com/new>.
- 3) Introduce un nombre para el repositorio y una descripción opcional.
- 4) Selecciona si quieres que sea público (visible para todos) o privado (sólo accesible para personas que invites).
- 5) Opcionalmente, puedes inicializarlo con un archivo README
- 6) Haz clic en "Create repository".

- ¿Cómo crear una rama en Git?

Las ramas en Git permiten trabajar en diferentes versiones del código sin afectar la principal. Para crear una nueva rama, usa:

```
$ git branch nombre_rama
```

Esto creará la rama, pero seguirás en la actual.

- ¿Cómo cambiar a una rama en Git?

Para moverte a otra rama, usa:

```
$ git checkout nombre_rama
```

O, en versiones recientes de Git:

```
$ git switch nombre_rama
```

Este comando cambia tu espacio de trabajo a la nueva rama.

- ¿Cómo fusionar ramas en Git?

Cuando terminas de trabajar en una rama, puedes fusionarla con la principal (por lo general "main"). Para hacerlo:

Cambia a la rama principal:

```
$ git checkout main o git switch main
```

Fusiona la otra rama:

```
$ git merge nombre_rama
```

- ¿Cómo crear un commit en Git?

Un commit guarda los cambios en tu repositorio local. Para hacerlo:

Agrega los cambios al área de preparación (staging):

```
$ git add .
```

Crea el commit con un mensaje descriptivo:

```
$ git commit -m "Descripción breve del cambio"
```

- ¿Cómo enviar un commit a GitHub?

Una vez que tienes cambios guardados en tu repositorio local, debes enviarlos a GitHub:

Asegúrate de tener un repositorio remoto vinculado:

```
$ git remote add origin URL_DEL_REPOSITORIO
```

Empuja los cambios a la rama deseada:

```
$ git push origin nombre_rama
```

- ¿Qué es un repositorio remoto?

Es un repositorio almacenado en un servidor (como GitHub) que permite que múltiples personas trabajen en el mismo proyecto desde diferentes ubicaciones.

- ¿Cómo agregar un repositorio remoto a Git?

Si tienes un proyecto local y quieres enlazarlo con un repositorio en GitHub, usa:

```
$ git remote add origin URL_DEL_REPOSITORIO
```

- ¿Cómo empujar cambios a un repositorio remoto?

Para enviar los cambios guardados localmente a GitHub:

```
$ git push origin nombre_rama
```

- ¿Cómo tirar de cambios de un repositorio remoto?

Para actualizar tu copia local con los cambios de GitHub:

```
$ git pull origin nombre_rama
```

- ¿Qué es un fork de repositorio?

Un fork es una copia de un repositorio que te permite hacer cambios sin afectar el original. Es útil cuando quieres contribuir a proyectos abiertos.

- ¿Cómo crear un fork de un repositorio?

- 1) Ve al repositorio en GitHub.
- 2) Haz clic en "Fork" en la parte superior derecha.
- 3) Espera a que se cree la copia en tu cuenta.

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

- 1) En el fork, sube tus cambios.
- 2) Ve a "Pull Requests" en el repositorio original.
- 3) Haz clic en "New pull request".
- 4) Selecciona la rama con tus cambios.
- 5) Agrega una descripción y envía la solicitud.

- ¿Cómo aceptar una solicitud de extracción?

- 1) Revisa los cambios en "Pull Requests".
- 2) Haz clic en "Merge pull request".
- 3) Confirma la fusión y elimina la rama si es necesario.

- ¿Qué es una etiqueta en Git?

Una etiqueta (tag) en Git es una referencia a un punto específico en la historia de un proyecto, utilizada comúnmente para marcar versiones importantes del software, como lanzamientos oficiales. A diferencia de las ramas, las etiquetas no cambian con el tiempo, lo que las hace ideales para marcar hitos.

- ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta anotada, que incluye un mensaje y metadatos adicionales, usa el siguiente comando:

```
$ git tag -a v1.0 -m "Versión 1.0"
```

Si solo necesitas una etiqueta ligera (sin metadatos), usa:

```
$ git tag v1.0
```

- ¿Cómo enviar una etiqueta a GitHub?

Para subir todas las etiquetas locales a GitHub, usa:

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

Si solo deseas subir una etiqueta específica, usa:

```
$ git push origin v1.0
```

- ¿Qué es un historial de Git?

El historial de Git es un registro de todos los commits realizados en un proyecto.

Para visualizarlo, usa:

```
$ git log
```

Para ver un resumen compacto:

```
$ git log --oneline --graph --decorate --all
```

- ¿Cómo buscar en el historial de Git?

Para buscar commits que contengan una palabra clave en su mensaje:

```
$ git log --grep="término"
```

Para buscar por autor:

```
$ git log --author="nombre"
```

- ¿Cómo borrar el historial de Git?

Si necesitas resetear los commits recientes, usa:

`$ git reset --hard HEAD~n` # Reemplaza 'n' con el número de commits a eliminar

- ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un repositorio que solo es accesible para el propietario y las personas a las que se les otorgan permisos. Es útil para proyectos que requieren confidencialidad o para desarrollo interno.

- ¿Cómo crear un repositorio privado en GitHub?

- 1) Ve a <https://github.com/new>.
- 2) Introduce el nombre del repositorio y una descripción opcional.
- 3) Selecciona la opción "Private".
- 4) Configura el archivo README.
- 5) Haz clic en "Create repository".

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

- 1) Ve a la pestaña "Settings" de tu repositorio.
- 2) Selecciona "Manage Access" en el menú lateral.
- 3) Haz clic en "Invite a collaborator".
- 4) Introduce el nombre de usuario o correo de quien vayas a invitar y envía la invitación.

- ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio visible para cualquier usuario de la plataforma. Cualquier persona puede verlo y, dependiendo de la configuración, puede contribuir a él.

- ¿Cómo crear un repositorio público en GitHub?

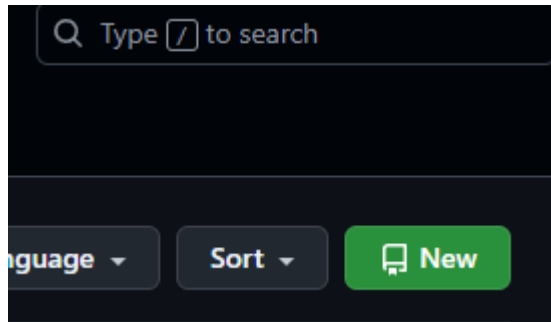
Sigue los mismos pasos que al crear un repositorio privado, pero selecciona la opción "Public" en lugar de "Private".

- ¿Cómo compartir un repositorio público en GitHub?

Puedes compartir el enlace del repositorio con cualquier persona. También puedes permitir contribuciones configurando issues y pull requests en la pestaña "Settings" del repositorio.

2) Realizar la siguiente actividad:

- Crear un repositorio.



- o Dale un nombre al repositorio.
- o Elije el repositorio sea público.
- o Inicializa el repositorio con un archivo.

- Agregando un Archivo
 - o Crea un archivo simple, por ejemplo, "mi-archivo.txt".
 - o Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.
 - o Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).

```
User@DESKTOP-NN2RFI8 MINGW64 ~/OneDrive/Desktop/Pre_Uni/P1
$ git clone https://github.com/Usinarock71/TestRepositorio.git
Cloning into 'TestRepositorio'...
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.
```

```
User@DESKTOP-NN2RFI8 MINGW64 ~/OneDrive/Desktop/Pre_Uni/
$ git 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:   text-example.txt

User@DESKTOP-NN2RFI8 MINGW64 ~/OneDrive/Desktop/Pre_Uni/
$ git commit -m "Agregando un archivo .txt"
[main f61f7a5] Agregando un archivo .txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 text-example.txt
```

```
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 305 bytes | 305.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Usinarock71/TestRepositorio.git
cd74bcd..f61f7a5  main -> main
```


Usinarock71 Agregando un archivo .txt
 f61f7a5 · 2 minutes ago
🕒 2 Commits

 README.md	Initial commit	6 minutes ago
 text-example.txt	Agregando un archivo .txt	2 minutes ago


README


TestRepositorio

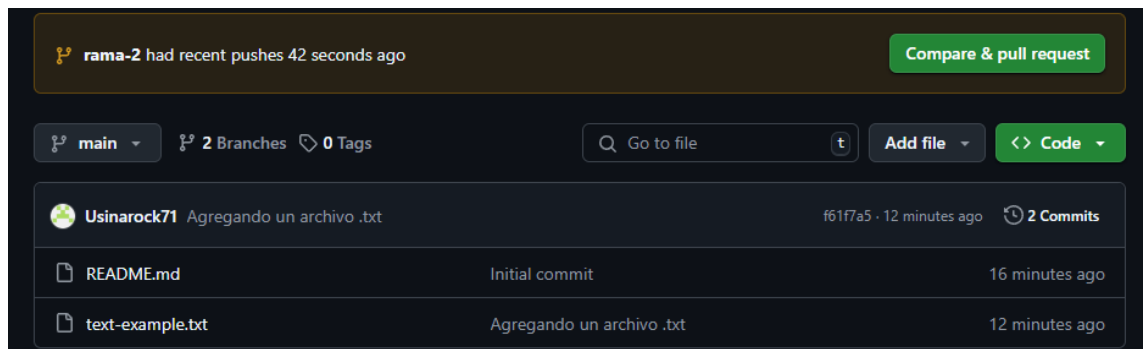
- Creando Branchs
 - o Crear una Branch
 - o Realizar cambios o agregar un archivo
 - o Subir la Branch

```
$ git checkout -b rama-2
Switched to a new branch 'rama-2'
```

```
$ git status
On branch rama-2
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   text-example.txt
```

```
$ git commit -m "Agregando modificacion desde rama-2"
[rama-2 c6d220c] Agregando modificacion desde rama-2
 1 file changed, 1 insertion(+)

User@DESKTOP-NN2RFI8 MINGW64 ~/OneDrive/Desktop/Pre_Uni/P1/TestRepositorio (rama-2)
$ git push origin rama-2
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 344 bytes | 344.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'rama-2' on GitHub by visiting:
remote:   https://github.com/Usinarock71/TestRepositorio/pull/new/rama-2
remote:
To https://github.com/Usinarock71/TestRepositorio.git
 * [new branch]      rama-2 -> rama-2
```



The screenshot shows the GitHub web interface for a repository named 'TestRepositorio'. At the top, a notification bar indicates that the 'rama-2' branch had recent pushes 42 seconds ago, with a 'Compare & pull request' button. Below this, the repository's main branch is set to 'main', and there are 2 branches and 0 tags. A search bar and buttons for 'Add file' and 'Code' are visible. The commit history shows two commits by user 'Usinarock71':

Commit Hash	Commit Message	Time Ago
f61f7a5	Agregando un archivo .txt	12 minutes ago
(Previous)	Initial commit	16 minutes ago

3) 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`
- Entra en el directorio del repositorio:

`cd conflict-exercise`

```
User@DESKTOP-NN2RFI8 MINGW64 ~/OneDrive/Desktop/Pre_Uni/P1
$ git clone https://github.com/Usinarock71/conflict-exercise.-.git
Cloning into 'conflict-exercise.-'...
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.

User@DESKTOP-NN2RFI8 MINGW64 ~/OneDrive/Desktop/Pre_Uni/P1
$ cd conflict-exercise.-

User@DESKTOP-NN2RFI8 MINGW64 ~/OneDrive/Desktop/Pre_Uni/P1/conflict-exercise.- (main)
$
```

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:

```
git checkout -b 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.

- Guarda los cambios y haz un commit:

```
git add README.md
```

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

```
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
```

```
1  # conflict-exercise.-
2  Este es un cambio en la feature branch.
```

```
$ git commit -m "Added a line in feature-branch"
[feature-branch ccd7494] Added a line in feature-branch
1 file changed, 2 insertions(+), 1 deletion(-)
```

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

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

```
git checkout main
```

- Edita el archivo README.md de nuevo, añadiendo una línea diferente:

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"
```

```
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

```
1 # conflict-exercise.-
2 Este es un cambio en la main branch. |
```

```
$ git add README.md

User@DESKTOP-NN2RFI8 MINGW64 ~/OneDrive/Desktop/
main)
$ git commit -m "Added a line in main branch"
[main 856c354] Added a line in main branch
1 file changed, 2 insertions(+), 1 deletion(-)
```

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

```
$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

Paso 6: Resolver el conflicto

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

```
<<<<<<< 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(Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se

debe borrar la parte del texto que no se quiera dejar).

- Añade el archivo resuelto y completa el merge:

```
git add README.md
```

```
git commit -m "Resolved merge conflict"
```

```
# conflict-exercise.-
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
Este es un cambio en la main branch.
=====
Este es un cambio en la feature branch.
>>>>>>> feature-branch (Incoming Change)
```

Resolucion:

```
# conflict-exercise.-
Este es un cambio en la main branch.
Este es un cambio en la feature branch.
```

```
$ git add README.md

User@DESKTOP-NN2RFI8 MINGW64 ~/OneDrive/De
main|MERGING)
$ git commit -m "Resolved merge conflict"
[main 7812fd6] Resolved merge conflict
```

Paso 7: Subir los cambios a GitHub

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

```
git push origin main
```

- También sube la feature-branch si deseas:

```
git push origin feature-branch
```

```
$ 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), 893 bytes | 446.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Usinarock71/conflict-exercise.-.git
7fff5ad..7812fd6 main -> main
```

```
$ 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/Usinarock71/conflict-exercise.-/pull/new/feature-branch
remote:
To https://github.com/Usinarock71/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.

