

Trabajo Práctico 2: Git y GitHub

1)

- ¿Qué es GitHub?

GitHub es una plataforma en línea que permite almacenar proyectos utilizando Git. Los repositorios se guardan en la nube, ya sea de forma pública o privada, lo que permite compartir y colaborar con la comunidad, fomentando el trabajo colaborativo entre desarrolladores. Además, incluye herramientas importantes para la organización y revisión de código, lo que la convierte en una solución integral para el desarrollo de software.

- ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en GitHub, se debe crear una cuenta o iniciar sesión en una preexistente. Luego, se presiona el botón de “+” que se encuentra en la parte superior derecha, y se hace click en “Nuevo repositorio”. Se ingresa el nombre del repositorio, se elige si será público o privado y finalmente se presiona el botón de “Crear repositorio”.

- ¿Cómo crear una rama en Git?

Para crear una rama en Git, se utiliza el comando “git branch **nombre-rama**”.

- ¿Cómo cambiar a una rama en Git?

Para cambiar a una rama en Git se utiliza el comando “git checkout **nombre-rama**”

- ¿Cómo fusionar ramas en Git?

Para fusionar ramas en Git se debe posicionar en la rama que queremos integrar los cambios, luego se utiliza el comando “git merge **nombre-rama**”.

- ¿Cómo crear un commit en Git?

Para crear un commit en Git, debemos utilizar el comando “git commit -m “**mensaje descriptivo**””

- ¿Cómo enviar un commit a GitHub?

Para enviar un commit a GitHub se deben realizar los siguientes pasos:

- Crear un repositorio
- Copiar la URL del repositorio
- En el proyecto local, utilizar el comando: “git remote add origin **URL**” y pegar la URL en la parte que lo especifica.
- Enviar el commit al repositorio remoto con el comando: “git push origin **main (o el nombre de la rama correspondiente)**.”

- ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión del proyecto, que está almacenado en la nube y que permite sincronizar el trabajo con otros desarrolladores.

- ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git se utiliza el comando “git remote add origin [URL](#)”

- ¿Cómo empujar cambios a un repositorio remoto?

Para empujar cambios a un repositorio remoto se utiliza el comando “git push origin [nombre-de-la-rama](#)”

- ¿Cómo tirar de cambios de un repositorio remoto?

Para tirar o traer cambios de un repositorio remoto se utiliza el comando “git pull origin [nombre-de-la-rama](#)”

- ¿Qué es un fork de repositorio?

Un fork es una copia completa de un repositorio original de GitHub, que se crea dentro de la propia cuenta en esta plataforma. Permite contribuir, mejorar y probar cambios en un proyecto externo, sin modificar el proyecto original.

- ¿Cómo crear un fork de un repositorio?

Primero se debe elegir el repositorio, luego se debe hacer click en el botón “Fork”. Elegiremos el nombre del nuevo repositorio, agregaremos una breve descripción y elegimos qué ramas copiar. Para finalizar apretaremos el botón de “Crear fork”.

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

Primero se debe elegir el repositorio al cuál ya le hicimos fork. Luego, presionaremos el botón de “Contribuir” y hacemos click en “Abrir pull request”. Elegimos la rama que deseamos, agregamos un título y descripción de los cambios realizados. Para finalizar, hacemos click en “Crear pull request”.

- ¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción, primero se debe ingresar a la pestaña “Pull requests” del repositorio en GitHub. Luego, se selecciona la solicitud recibida, se revisan los cambios propuestos y, si están correctos, se hace clic en “Merge pull request”. Por último, se confirma la acción con “Confirm merge” para que los cambios se integren al proyecto principal.

- ¿Qué es una etiqueta en Git?

Una etiqueta en Git, sirve para marcar una versión específica y estable de un proyecto. Es útil para identificar lanzamientos, facilitar retrocesos y organizar el desarrollo.

- ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta en Git utilizamos el comando “git tag [nombre-etiqueta](#)”

- ¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta a GitHub se debe utilizar el comando “git push origin [nombre-etiqueta](#)”.

- ¿Qué es un historial de Git?

El historial de Git es el registro completo de todos los commits (cambios confirmados) realizados. Cada modificación queda registrada junto con el autor, la fecha y un mensaje explicativo. Esto permite auditar los cambios, rastrear errores o incluso deshacer modificaciones si es necesario.

- ¿Cómo ver el historial de Git?

Para ver el historial de Git se utiliza el comando “git log”.

- ¿Cómo buscar en el historial de Git?

Para buscar en el historial de Git se usa “git log” con opciones como [--grep](#) (para palabras clave), [--author](#) (por autor) o [-S](#) (por contenido modificado). Esto permite encontrar cambios específicos de forma rápida.

- ¿Cómo borrar el historial de Git?

Se puede borrar la carpeta “.git”, que elimina por completo el historial y configuración del repositorio. El proyecto deja de estar bajo control de versiones. Solo se recomienda hacerlo si se desea reiniciar el repositorio desde cero.

- ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub, es aquel que solo puede ser visto y modificado por los dueños de ese repositorio o las personas que él elija.

- ¿Cómo crear un repositorio privado en GitHub?

Para crear un repositorio privado en GitHub, solo se debe seleccionar la opción de “Private”.

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

Para invitar a alguien a un repositorio privado en GitHub, se debe presionar el botón “Invitar colaboradores”, luego buscarlos en la lupa y agregarlos/invitarlos al repositorio.

- ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio al cual todos pueden acceder, realizar fork y pull request.

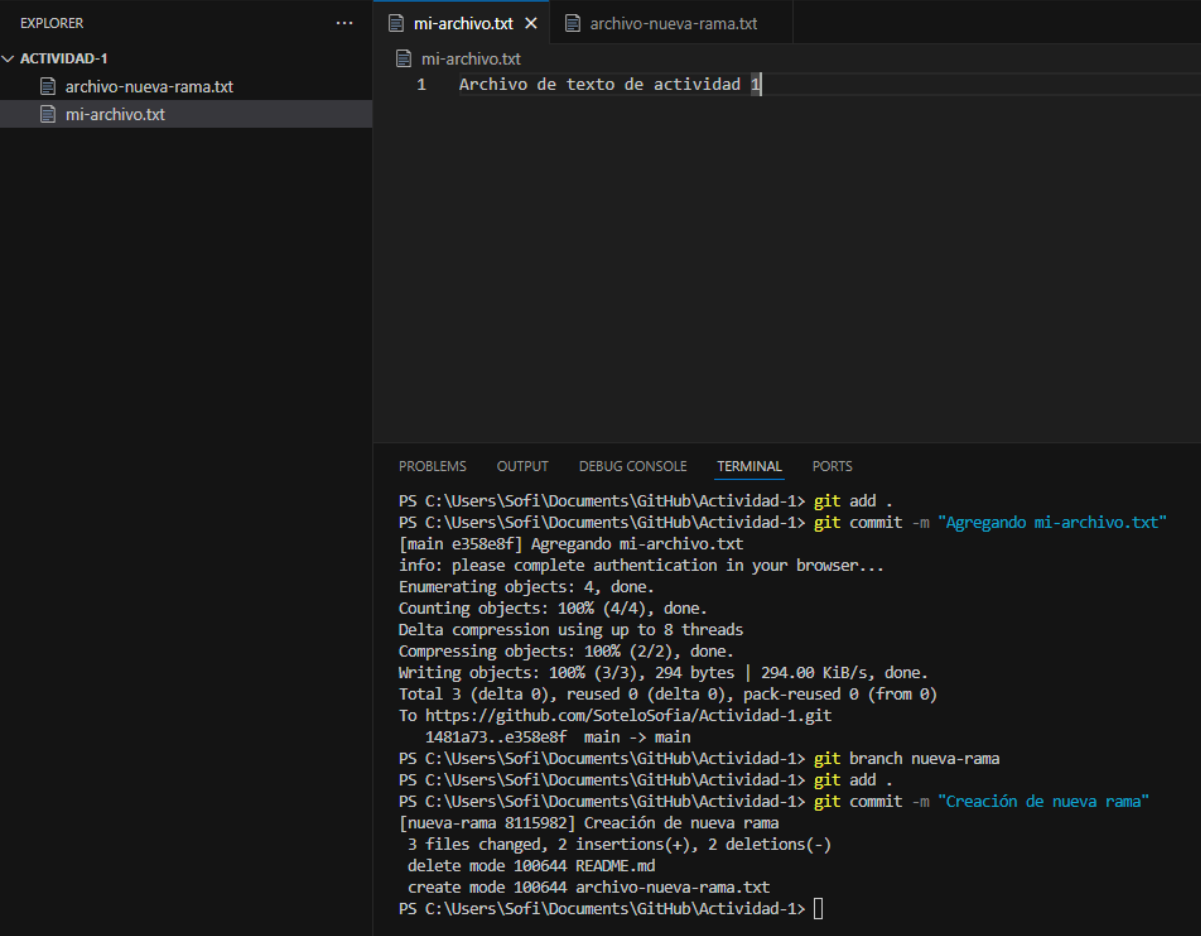
- ¿Cómo crear un repositorio público en GitHub?

Para crear un repositorio público en GitHub, solo se debe seleccionar la opción de "Public".

- ¿Cómo compartir un repositorio público en GitHub?

Para compartir un repositorio público en GitHub, se debe copiar la URL del repositorio y compartirla.

2)



The screenshot shows a Visual Studio Code interface. On the left, the Explorer pane shows a folder named 'ACTIVIDAD-1' containing two files: 'archivo-nueva-rama.txt' and 'mi-archivo.txt'. The main editor area shows 'mi-archivo.txt' with the content '1 Archivo de texto de actividad 1'. At the bottom, the TERMINAL pane displays the following commands and output:

```
PS C:\Users\Sofi\Documents\GitHub\Actividad-1> git add .
PS C:\Users\Sofi\Documents\GitHub\Actividad-1> git commit -m "Agregando mi-archivo.txt"
[main e358e8f] Agregando mi-archivo.txt
info: please complete authentication in your browser...
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 294 bytes | 294.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/SoteloSofia/Actividad-1.git
 1481a73..e358e8f main -> main
PS C:\Users\Sofi\Documents\GitHub\Actividad-1> git branch nueva-rama
PS C:\Users\Sofi\Documents\GitHub\Actividad-1> git add .
PS C:\Users\Sofi\Documents\GitHub\Actividad-1> git commit -m "Creación de nueva rama"
[nueva-rama 8115982] Creación de nueva rama
3 files changed, 2 insertions(+), 2 deletions(-)
delete mode 100644 README.md
create mode 100644 archivo-nueva-rama.txt
PS C:\Users\Sofi\Documents\GitHub\Actividad-1>
```

<https://github.com/SoteloSofia/Actividad-1.git>

3)

```
PS C:\Users\Sofi\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Sofi\conflict-exercise> git commit -m "Added a line in feature-branch"
[feature-branch 1cdb384] Added a line in feature-branch
 1 file changed, 1 insertion(+)
PS C:\Users\Sofi\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Sofi\conflict-exercise> git add README.md
PS C:\Users\Sofi\conflict-exercise> git commit -m "Added a line in main branch"
[main 8e6f749] Added a line in main branch
 1 file changed, 1 insertion(+)
PS C:\Users\Sofi\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.
PS C:\Users\Sofi\conflict-exercise> █
```

```
README.md > # Actividad 3<<<<<< HEADEste es un cambio en la main branch.
```

```
1 # conflict-exercise
2 Actividad 3
  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
3 <<<<<< HEAD (Current Change)
4 Este es un cambio en la main branch.
5 =====
6 Este es un cambio en la feature branch.
7 >>>>>> feature-branch (Incoming Change)
8
```

```
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
PS C:\Users\Sofi\conflict-exercise> git add README.md
PS C:\Users\Sofi\conflict-exercise> git commit -m "Resolved merge conflict"
[main 9976d67] Resolved merge conflict
PS C:\Users\Sofi\conflict-exercise> █
```

```
PS C:\Users\Sofi\conflict-exercise> git commit -m "Resolved merge conflict"
[main 9976d67] Resolved merge conflict
PS C:\Users\Sofi\conflict-exercise> git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 857 bytes | 285.00 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/SoteloSofia/conflict-exercise.git
 581cac3..9976d67  main -> main
PS C:\Users\Sofi\conflict-exercise> █
```

<https://github.com/SoteloSofia/conflict-exercise.git>