# Práctico 2: Git y GitHub

## Objetivo:

El estudiante desarrollará competencias para trabajar con Git y GitHub, aplicando conceptos fundamentales de control de versiones, colaboración en proyectos y resolución de conflictos, en un entorno simulado y guiado.

*Resultados de aprendizaje:*

1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.
2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto.
3. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo.
4. Resolver conflictos en un entorno de control de versiones: Identificar, analizar y solucionar conflictos de merge generados en un flujo de trabajo con múltiples ramas.

*Actividades*

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

- ¿Qué es GitHub?

  *Github es una plataforma online que funciona como red social en la cual se guarda y comparte código y que permite el trabajo colaborativo entre desarrolladores de todo el mundo. Permite también que se registren cambios y estos puedan ser comparados con versiones anteriores.*

- ¿Cómo crear un repositorio en GitHub?

  *En github hay una opción de "Crear nuevo repositorio". Cuando hacemos esto debemos nombrarlo, agregarle una descripción (opcional), elegir si deseamos que sea público o privado, e inicializar un archivo README (opcional).*

- ¿Cómo crear una rama en Git?

  *Posicionados en el repositorio de nuestro proyecto, y habiendo inicializado git, abrimos GitBash y escribimos el comando "git branch" y luego escribir el nombre de la nueva rama a crear.*

- ¿Cómo cambiar a una rama en Git?

  *Utilizando el comando "git checkout" seguido por el nombre de la rama a la cual queremos movernos.*

- ¿Cómo fusionar ramas en Git?

  *Es importante saber que debemos posicionarnos en la rama a la cual queremos*

*traer los cambios de otra. Si aún no estamos ahí, utilizamos el comando "git checkout" como se indicó antes. Una vez en la rama deseada, se escribe el comando "git merge" seguido por el nombre de la rama de la cual se traerán los cambios.*

- ¿Cómo crear un commit en Git?

  *Primero nos aseguramos que las modificaciones hechas pasen a estar "staged" con un "git add". Luego escribimos "git commit -m" seguido de un mensaje que identifique el commit. De esta forma los cambios serán guardados localmente.*

- ¿Cómo enviar un commit a GitHub?

  *Si se desea enviar un commit a Github para que los cambios no solo estén locales, se puede hacer "git push". Antes, se debe haber añadido el correspondiente repositorio remoto al que se enviaran los cambios.*

- ¿Qué es un repositorio remoto?

  *Es un repositorio de git guardado en un servidor o plataforma externa, como es el caso de Github. Es decir, es una copia de tu trabajo local guardada en otro lugar.*

- ¿Cómo agregar un repositorio remoto a Git?

  *Se debe utilizar el comando "git remote add origin" + url del repositorio. Es importante notar que origin es el nombre que se le da por convención al repositorio remoto.*

- ¿Cómo empujar cambios a un repositorio remoto?

  *Una vez agregado el repositorio remoto, se debe hacer un "git push" + repositorio remoto + rama actual. Por ejemplo, "git push origin main".*

- ¿Cómo tirar de cambios de un repositorio remoto?

  *Para traer cambios de un repositorio remoto al local se utiliza el comando "git pull" + repositorio remoto + rama actual. Por ejemplo, "git pull origin main".*

- *¿Qué es un fork de repositorio?*

  *Un fork es una creación de una copia de un repositorio de otros. En esta copia podemos experimentar y probar cambios en el código. Si queremos proponer que estos cambios se agreguen al proyecto original podemos solicitar un "pull request".*

- ¿Cómo crear un fork de un repositorio?

  *Para crear un fork tenemos que ir a un repositorio de otra cuenta de Github, seleccionar la opción "fork" y, de esta manera, se creara una copia del repositorio en nuestra cuenta de Github.*

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

  *Si querés que tus contribuciones sean agregadas a un repositorio original de cual hiciste un fork, podes ir al repositorio original y seleccionar la opción "pull requests" >> "nueva pull request". Ahí podés seleccionar la rama de tu fork y proponer los cambios deseados.*

- ¿Cómo aceptar una solicitud de extracción?

*Para aceptar una pull request tenés que ir a la pestaña de "pull requests" y abrir la solicitud de extracción que quieras revisar. Podes ver los cambios hechos y podés aceptar la solicitud y hacer un merge solucionando posibles conflictos.*

- ¿Qué es un etiqueta en Git?

  *Son nombres o alias que se le dan a commits a los cuales quiero identificar. Mediante el uso de ese alias luego podre referenciar ese commit. También sirve para identificar diferentes versiones de un programa.*

- ¿Cómo crear una etiqueta en Git?

  *Para crear una etiqueta utilizamos el comando "git tag" seguido del nombre de la etiqueta.*

- ¿Cómo enviar una etiqueta a GitHub?

  *Cuando haces un push las etiquetas no se envían al repositorio remoto. Para poder enviarlas hay que usar el comando "git push origin" + nombre de la etiqueta o, si quieres enviar todas las etiquetas, utilizamos el comando "git push --tags".*

- ¿Qué es un historial de Git?

  *Es un detalle de todos los commits del repositorio. Incluye los hash de cada commit, el autor, la fecha y los mensajes incluidos.*

- ¿Cómo ver el historial de Git?

  *Se puede ver el historial de Git con el comando "git log", de forma que recibiremos un historial completo con todos los detalles de los commits. Si se quiere ver solo los hash y los mensajes de cada commit (excluyendo la fecha y el autor), se puede utilizar el comando "git log --oneline".*

- ¿Cómo buscar en el historial de Git?

  *Dependiendo de qué se quiera buscar se utilizan varios comandos o filtros:*

  *Para buscar un commit especifico:*

  ```
  git show (hash del commit)
  ```

  *Para buscar por autor:*

  ```
  git log --author="……"
  ```

  *Para buscar por fecha:*

  ```
  git log --since="(año-mes-dia)" --until="(año-mes-dia)"
  ```

  *Para buscar cambios de un archivo en particular:*

  ```
  git log -- archivo.txt
  ```

  *Para buscar por palabras clave en el mensaje del commit:*

  ```
  git log --grep="(palabras)"
  ```

- ¿Cómo borrar el historial de Git?

  *Si se quiere eliminar los commits hechos con anterioridad pero se quieren mantener los archivos, se puede resetear la rama al primer commit y sobreescribir el historial*

*con los siguientes comandos:*

***- git checkout --orphan new-branch***

*(Esto crea una nueva rama (new-branch) sin ningún historial de commits anterior. Es como empezar un nuevo repositorio manteniendo los archivos actuales)*

***- git add -A***

*(Agrega todos los archivos (nuevos, modificados y eliminados) al área de preparación)*

***- git commit -m "Fresh start"***

*(Crea un nuevo primer commit en la rama huérfana)*

***- git branch -D main***

*(Elimina forzosamente la rama main existente. -D se usa en lugar de -d para borrar la rama incluso si no está completamente fusionada).*

***- git branch -m main***

*Renombra new-branch a main, convirtiéndola en la nueva rama principal.*

***- git push --force origin main***

*--force sobrescribe la rama main en el repositorio remoto con la nueva versión. Esto elimina todo el historial de commits anterior en el repositorio remoto.*

- ¿Qué es un repositorio privado en GitHub?

  *Un repositorio privado en Github es uno que no está visible al público. Solo su creador y usuarios con acceso explicito pueden acceder o contribuir al código.*

- ¿Cómo crear un repositorio privado en GitHub?

  *Cuando se crea un repositorio se elige que su visibilidad sea “privada” en vez de “pública”.*

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

  *Se puede compartir un repositorio privado cambiando la configuración de éste en la pestaña de configuración >> colaboradores y equipos. Allí ingresamos el usuario o email de la persona a la cual le queremos dar acceso, quien deberá aceptar la invitación para acceder al repositorio.*

- ¿Qué es un repositorio público en GitHub?

  *Un repositorio público, en cambio, es uno al cual se puede acceder solo teniendo su enlace, o buscandolo en github. Tambien podes descargarlo o hacer un fork. Es importante notar que aunque es visible para todos, no significa que puedas podificarlo sin permiso.*

- ¿Cómo crear un repositorio público en GitHub?

  *Cuando se crea un repositorio se elige que su visibilidad sea “pública” en vez de “privada”.*

- ¿Cómo compartir un repositorio público en GitHub?

*Podés compartirlo con su url o agregar colaboradores. Si queres que puedan modificar el código podes permitir que lo clonen o que hagan un fork y luego una pull request.*

2) Realizar la siguiente actividad:

- Crear un repositorio.
    - Dale un nombre al repositorio.
    - Elije el repositorio sea público.
    - Inicializa el repositorio con un archivo.
- 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).
- Creando Branchs
    - Crear una Branch
    - Realizar cambios o agregar un archivo
    - Subir la Branch

*Link al repositorio con el ejercicio resuelto: [https://github.com/Daniela-N-Romero/repo-actividad2/tree/main](https://github.com/Daniela-N-Romero/repo-actividad2/tree/main)*

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
```

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"
```

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"
```

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.

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"

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

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.

*Link al repositorio con el ejercicio resuelto:* https://github.com/Daniela-N-Romero/conflict-exerciseTP2 *. El primer intento me salió que no había conflicto porque modifique líneas diferentes. Luego puede crear el conflicto y ahí lo resolví como se pedía en el ejercicio.*