# 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? **-> Un servicio de alojamiento de repositorios de git**
- ¿Cómo crear un repositorio en GitHub? **-> Tras iniciar sesión dar click en el boton new/nuevo**
- ¿Cómo crear una rama en Git? **-> git branch *nombre***
- ¿Cómo cambiar a una rama en Git? **-> git checkout *nombre***
- ¿Cómo fusionar ramas en Git? **-> cambiar a la rama desde la cual se desea fusionar e ingresar git Merge *nombre***
- ¿Cómo crear un commit en Git? **-> tras realizar los cambios ingresar git add . y luego git commit**
- ¿Cómo enviar un commit a GitHub? **-> Luego de haber configurado el respositorio remoto ingresar git push *nombre***
- ¿Qué es un repositorio remoto? **-> Un repositorio alojado en otra computadora/servidor**
- ¿Cómo agregar un repositorio remoto a Git? **-> git remote add *nombre* *url***
- ¿Cómo empujar cambios a un repositorio remoto? **-> git push -u origin**
- ¿Cómo tirar de cambios de un repositorio remoto? **-> git pull –u origin**
- ¿Qué es un fork de repositorio? **-> Una copia de un repositorio donde se pueden aplicar cambios sin afectar al original**
- ¿Cómo crear un fork de un repositorio? **-> Desde la pagina principal del repositorio dar click en “Fork”**

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
- ¿Cómo aceptar una solicitud de extracción?
- ¿Qué es un etiqueta en Git? **-> Es un "marcador" que se usa para señalar versiones importantes, Ej el lanzamiento de una versión oficial (v1.0)**
- ¿Cómo crear una etiqueta en Git? **-> git tag *nombre***
- ¿Cómo enviar una etiqueta a GitHub? **-> git push *remoto* *tag* o git push *remote* --tags**
- ¿Qué es un historial de Git? **-> Donde se puede ver todos los cambios realizados desde la creación del repositorio**
- ¿Cómo ver el historial de Git? **-> git log**
- ¿Cómo buscar en el historial de Git? **-> Se puede buscar de distintas maneras, por palabras usando --grep=*palabra*, por autor usando –author=*nombre*, etc...**
  - ¿Cómo borrar el historial de Git? **-> El historial de git no puede ser borrado, para eso debería eliminarse el repositorio entero**
- ¿Qué es un repositorio privado en GitHub? **-> Un repositorio que solo el creado o las personas que se configuren pueden ingresar y modificar**
- ¿Cómo crear un repositorio privado en GitHub? **-> Luego darle al boton "new" se debe elegir la Opción "Private"**
- ¿Cómo invitar a alguien a un repositorio privado en GitHub? **-> Se debe ingresar al repositorio, opciones, colaboradores y ahi ingresar el mail de las personas a invitar.**
- ¿Qué es un repositorio público en GitHub? **-> Un repositorio al que todo el mundo puede acceder**
- ¿Cómo crear un repositorio público en GitHub? **-> Tras darle al boton "new" seleccionar la opcion "Public"**
- ¿Cómo compartir un repositorio público en GitHub? **-> Basta con compartir el enlace al repositorio**

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

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.