

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?**

Es un lugar donde se puede almacenar, compartir y colaborar en proyectos de programación utilizando un sistema llamado Git.

GitHub facilita este proceso proporcionando una interfaz amigable y muchas funciones adicionales, como la posibilidad de gestionar proyectos, revisar código, reportar problemas y publicar documentación. También es ampliamente utilizado para compartir proyectos de código abierto, donde cualquiera puede contribuir o usar el código compartido.

- **¿Cómo crear un repositorio en GitHub?**

Primero que nada, hay que acceder a la cuenta de GitHub o crear un usuario. Luego se hace click en "New" o "Nuevo Repositorio" que, generalmente está en la esquina derecha. Asignas un nombre a tu repositorio y agregas una descripción si así lo deseas. Puedes hacer que sea público o privado. Luego configuras el repositorio localmente si tienes el proyecto en la computadora.

- **¿Cómo crear una rama en Git?**

Para crear una rama en Git primero usamos el comando `git branch` para ver las ramas disponibles en el repositorio. Luego ejecutamos el comando `git branch "nombre-de-la-rama"` para crear una rama.

- **¿Cómo cambiar a una rama en Git?**

Para cambiar a la nueva rama usamos el comando `git checkout nombre-de-la-rama`.

- **¿Cómo fusionar ramas en Git?**

Para integrar cambios de diferentes ramas en una sola hay que cambiar a la rama principal (o la rama donde deseas fusionar) con el siguiente comando: `"git checkout main"`, luego colocamos `"git merge nombre-de la rama"` para fusionar la rama en la que has estado trabajando con la rama principal.

- **¿Cómo crear un commit en Git?**

Para crear un commit o "tomar una foto" del estado actual del proyecto en cuestión hay que utilizar el comando `git commit` con un mensaje que describa los cambios que se realizaron.

- **¿Cómo enviar un commit a GitHub?**

Para enviar los cambios al repositorio remoto hay que usar el comando `"git push"`. La primera vez que se realiza `"git push"` se pone la rama de origen. Luego basta solo con el `git push`.

- **¿Qué es un repositorio remoto?**

Es una versión del proyecto que está almacenada en un servidor en línea o en una red. Es donde se almacena todo lo que vamos subiendo.

- **¿Cómo agregar un repositorio remoto a Git?**

Primero hay que abrir la terminal del proyecto local, luego añadimos el repositorio remoto utilizando el comando `"git remote add"` seguido de un nombre y la URL del repositorio remoto. Por ejemplo: `git remote add origin`

https://github.com/TU_USUARIO/NOMNRE_REPOSITORIO.git. Para verificar que se haya añadido correctamente ejecutamos el comando `"git remote -v"`.

- **¿Cómo empujar cambios a un repositorio remoto?**

Utilizando el comando `"git push"` para enviar los cambios. Por ejemplo: `"git push origin main"`

- **¿Cómo tirar de cambios de un repositorio remoto?**

Cambias a la rama donde deseas aplicar los cambios con `"git checkout + el nombre-de-la-rama"`, ejecutas el comando `"git pull"` para traer los cambios desde el repositorio remoto. Ejemplo: `"git pull origin nombre-de-la-rama"`

- **¿Qué es un fork de repositorio?**

Un fork en Git es una copia de un repositorio que se crea en tu cuenta de usuario, generalmente en plataformas como GitHub.

- **¿Cómo crear un fork de un repositorio?**

Primero inicias sesión en la cuenta de GitHub, buscas el repositorio que deseas forkear y haces click en el botón fork que se encuentra en la esquina superior derecha de la página del repositorio, seleccionas tu cuenta como destino y una vez creado serás redirigido a tu copia del repositorio en tu cuenta.

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

Te diriges a la página principal del repositorio. En el menú "Rama", selecciona la rama que contiene tus confirmaciones. Sobre la lista de archivos, en el banner amarillo, haz clic en "Comparar y solicitud de extracción" para crear una solicitud de extracción para la rama asociada.

- **¿Cómo aceptar una solicitud de extracción?**

Vas a la pestaña "pull request" en el repositorio de GitHub, haces click en la solicitud que deseas revisar, examinas los cambios propuestos, los archivos modificados y los comentarios si los hubiera. Opcional: puedes descargar la rama de la solicitud de extracción localmente para probar los cambios antes de aceptarlos. Por ejemplo: "git fetch origin nombre-de-la-rama" "git checkout nombre-de-la-rama". Resuelves conflictos si los hay, fusiona la solicitud de extracción, haces click en el botón "Merge pull request" y confirmas la fusión haciendo click en "Confirm merge". También puedes eliminar la rama de la solicitud si ya no fuese necesaria.

- **¿Qué es una etiqueta en Git?**

Es una referencia que marca un punto específico en el historial de commits. Se usa comúnmente para señalar versiones importantes del proyecto, como lanzamientos oficiales. A diferencia de las ramas, no cambian con el tiempo, lo que las hace ideales para identificar versiones estables.

- **¿Cómo crear una etiqueta en Git?**

Primero asegurarse de estar en la rama correcta con el git checkout, luego creas una etiqueta ligera con el comando "git tag "nombre-de-la-etiqueta"" o una etiqueta anotada con "git tag -a nombre-de-la-etiqueta -m

- **¿Cómo enviar una etiqueta a GitHub?**

Con el comando "git push origin + nombre-de-la-etiqueta"

- **¿Qué es un historial de Git?**

Es el registro de todos los cambios realizados en un repositorio a lo largo del tiempo. Cada vez que realizas un commit, Git guarda una "instantánea" del estado del proyecto, permitiendo que puedas ver qué modificaciones se han hecho, quién las hizo y cuándo ocurrieron.

- **¿Cómo ver el historial de Git?**

Con el comando "git log", que muestra una lista de commits en orden cronológico inverso, con detalles como el identificador único del commit, el autor, la fecha y el mensaje asociado.

- **¿Cómo buscar en el historial de Git?**

Buscando commits que contengan una palabra clave: "git log --grep="palabra clave". También puedes buscar cambios en un archivo específico "git log -- nombre-del-archivo", mostrará todos los commits que han modificado el archivo. O ver qué commits afectaron una línea específica: "git blame nombre-del-archivo" muestra quien modificó cada línea y en qué commit. O buscar dentro del código fuente en cualquier commit: "git grep "texto a buscar" que buscará la cadena dentro de todos los archivos del repositorio.

- **¿Cómo borrar el historial de Git?**

Con los siguientes comandos :

```
git init
```

```
git add .
```

```
git commit -m "Nuevo inicio del repositorio"
```

```
git remote add origin https://github.com/TU_USUARIO/NOMBRE_REPOSITORIO.git
```

```
git push -f origin main, los cuales eliminarán todo el historial y reiniciarán el repositorio
```

- **¿Qué es un repositorio privado en GitHub?**

Es un repositorio que sólo es accesible para los propietarios y colaboradores autorizados. A diferencia de los repositorios públicos, donde cualquier persona puede ver el código, los repositorios privados permiten mantener el contenido confidencial o restringido a un equipo específico

- **¿Cómo crear un repositorio privado en GitHub?**

Haces click en el botón "New", asignas un nombre a tu repositorio, seleccionas la opción "private" y lo puedes inicializar con un archivo README, .gitignore o una licencia.

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

Si deseas compartir el repositorio con colaboradores, ve a Settings > Collaborators y agrega sus nombres de usuario.

- **¿Qué es un repositorio público en GitHub?**

Un repositorio público en GitHub es un repositorio accesible para cualquier persona en internet. Esto significa que cualquier usuario puede ver el código, descargarlo y, dependiendo de la configuración del repositorio, incluso contribuir a él mediante pull requests.

- **¿Cómo crear un repositorio público en GitHub?**

Haces click en el botón “New”, asignas un nombre a tu repositorio, seleccionas la opción “public” y lo puedes inicializar con un archivo README, .gitignore o una licencia.

- **¿Cómo compartir un repositorio público en GitHub?**

Accedes al repositorio, copias el enlace del mismo que se encuentra en la parte superior con un botón llamado “Code”, clickeas y copias la URL que aparece para compartirla con quien desees.

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).

```
MINGW64:/c/Users/gpais/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAM...
$ git init
Reinitialized existing Git repository in C:/Users/gpais/Desktop/UTN - TUP/Primer
cuatrimestre/PROGRAMACIÓN I/Git/tp 2/TP-2-P1/.git/

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/TP-2-P1 (main)
$ git add .

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/TP-2-P1 (main)
$ git commit -m "Subiendo los archivos"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/TP-2-P1 (main)
$ git push origin main
Everything up-to-date

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/TP-2-P1 (main)
$
```

- 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".

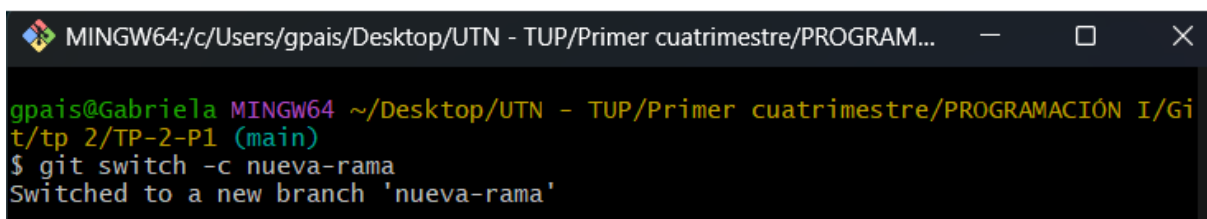
Paso 2: Clonar el repositorio a tu máquina local

- 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
```



```
MINGW64:/c:/Users/gpais/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2/TP-2-P1 (main)
$ git switch -c nueva-rama
Switched to a new branch 'nueva-rama'
```



```
gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2/TP-2-P1 (nueva-rama)
$ git commit -m "creamos nueva rama y la modificamos creando un nuevo archivo dentro"
On branch nueva-rama
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        nuevo-archivo.txt

nothing added to commit but untracked files present (use "git add" to track)

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2/TP-2-P1 (nueva-rama)
$ git add .

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2/TP-2-P1 (nueva-rama)
$
```

```
MINGW64:/c:/Users/gpais/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAM...
gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2 (main)
$ git clone https://github.com/GitGPais/conflict-exercise.git
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.

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2 (main)
$ cd conflict-exercise

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2/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"
```

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

```
gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
```

```
MINGW64:/c/Users/gpais/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2/conflict-exercise (feature-branch)
gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2/conflict-exercise (feature-branch)
git add README.md

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2/conflict-exercise (feature-branch)
git commit -m "Added a line in feature-branch"
feature-branch b5b6059] Added a line in feature-branch
1 file changed, 1 insertion(+)

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2/conflict-exercise (feature-branch)
|
```

- 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.

```
MINGW64:/c/Users/gpais/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/conflict-exercise (main)
$ git add README.md

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/conflict-exercise (main)
$
```

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.



conflict-exercise / README.md

GitGPais Update README.md d94511c · 38 minutes ago History

Preview Code Blame 7 lines (4 loc) · 142 Bytes Raw Copy Download Edit

conflict-exercise

##Repo para generar un merge conflict

##Este es un cambio en la feature branch

##Este es un cambio en la main branch.

History for conflict-exercise / README.md on main

All users All time

Commits on Apr 5, 2025

Update README.md GitGPais authored 38 minutes ago	Verified d94511c Copy Download <>
Merge branch 'feature-branch' GitGPais committed 43 minutes ago	9c57d73 Copy Download <>
Added a line in main branch GitGPais committed 45 minutes ago	6545f15 Copy Download <>
Este es un cambio en la feature branch GitGPais committed 52 minutes ago	d6314a2 Copy Download <>
Added a line in feature-branch GitGPais committed 1 hour ago	b5b6059 Copy Download <>
Initial commit GitGPais authored 1 hour ago	Verified 52b163d Copy Download <>

MINGW64:/c/Users/gpais/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Git/tp 2/Nueva carpeta/conflict-exercise

```
gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/Nueva carpeta/conflict-exercise (main)
$ git add README.md

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/Nueva carpeta/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 6545f15] Added a line in main branch
1 file changed, 1 insertion(+)

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/Nueva carpeta/conflict-exercise (main)
$ 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.

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/Nueva carpeta/conflict-exercise (main|MERGING)
$ git add README.md

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/Nueva carpeta/conflict-exercise (main)
$ git commit -m "Resolved merge conflict"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/Nueva carpeta/conflict-exercise (main)
$ git push origin main
Everything up-to-date

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/Nueva carpeta/conflict-exercise (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/GitGPais/conflict-exercise/pull/new/feature-bran
ch
remote:
remote: To https://github.com/GitGPais/conflict-exercise.git
* [new branch]   feature-branch -> feature-branch

gpais@Gabriela MINGW64 ~/Desktop/UTN - TUP/Primer cuatrimestre/PROGRAMACIÓN I/Gi
t/tp 2/Nueva carpeta/conflict-exercise (main)
$
```