

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,
3. fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto.
4. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo.
5. 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.

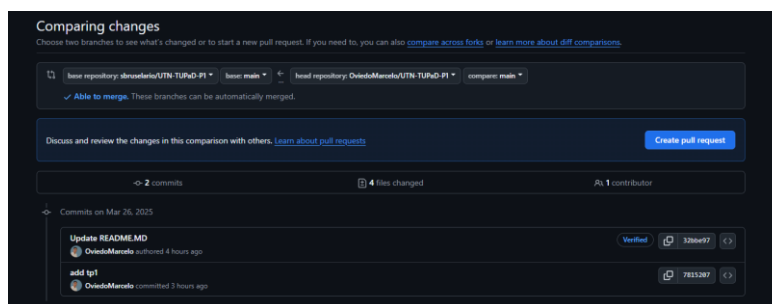
Actividades

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

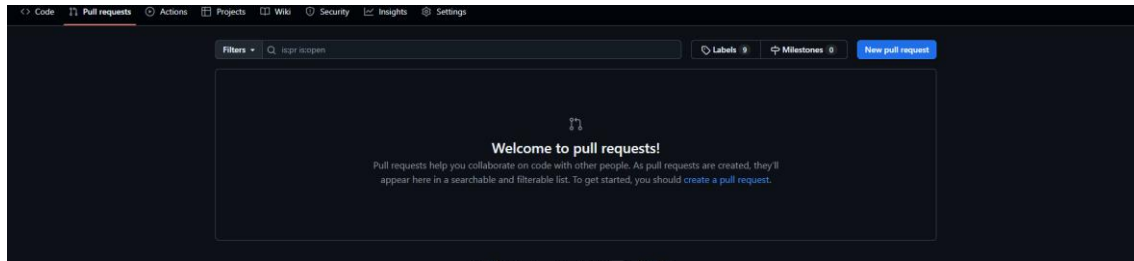
(Desarrollar las respuestas):

- ¿Qué es GitHub? Github es un repositorio en la nube para poder subir un repositorio de git local y poder utilizarlo desde donde sea, no solo de forma local. Además permite trabajar con otras personas en conjunto en el mismo repositorio.
- ¿Cómo crear un repositorio en GitHub? Lo podemos crear desde la web de github, una vez logueados en nuestra cuenta, dirigiéndonos a “repositories” → “new” y completando las opciones. También podemos hacerlo por consola una vez logueada nuestra cuenta desde el git bash *gh auth login* ejecutamos el siguiente comando: *gh repo create nombre-del-repo --public*
- ¿Cómo crear una rama en Git? Para crear una rama en git debemos ejecutar en consola *git branch nombre-de-la-rama*
- ¿Cómo cambiar a una rama en Git? Para cambiar de una rama a otra rama debemos ejecutar en consola *git checkout nombre-de-la-rama-destino*

- ¿Cómo fusionar ramas en Git? Para fusionar ramas en git debemos, una vez parado en la rama que queremos fusionar *git merge rama-dev* fusionando así en la rama actual los cambios que existen en *rama-dev*
- ¿Cómo crear un commit en Git? Para crear un commit en git debemos ejecutar *git commit -m "detalle del commit"* siendo -m una opción para dejar un detalle de que se hizo en este nuevo commit. Previamente debemos agregar todos los cambios a ese commit ejecutando *git add*.
- ¿Cómo enviar un commit a GitHub? Para enviar un commit local al repositorio de github debemos ejecutar *git push*
- ¿Qué es un repositorio remoto? Un repositorio remoto es un lugar donde podemos almacenar todos los archivos de un proyecto para que varias personas o nosotros solos, podamos acceder y subir cambios desde cualquier sitio que estemos.
- ¿Cómo agregar un repositorio remoto a Git? Para agregar un repositorio remoto a git primero debemos inicializarlo con el comando *git init* y una vez inicializado vincularemos el repositorio remoto que tengamos generado con el comando *git remote add origin "url_de_el_repositorio_remoto"*
- ¿Cómo empujar cambios a un repositorio remoto? Una vez realizado el commit en git debemos enviar los cambios ejecutando en la consola *git push*
- ¿Cómo tirar de cambios de un repositorio remoto? Para obtener los cambios realizado en el repositorio remoto debemos ejecutar el comando *git pull* lo que nos va a traer todos los cambios en la rama en la que estemos parados del repo remoto.
- ¿Qué es un fork de repositorio? Hacer un fork de un repositorio remoto es copiar completamente un repositorio de otros usuario y llevarlo a un repositorio nuevo propio de nuestra cuenta.
- ¿Cómo crear un fork de un repositorio? Para crear un fork de un repositorio podemos hacerlo de 2 formas, dirigiéndonos en la web del repositorio que queremos copiar y seleccionar la opción "fork" completando la información necesaria básica antes de llevarlo a un repositorio propia. Si tenemos alguna interfaz de usuario como GitHub CLI podríamos hacerla desde ahí con el comando *gh repo fork usuario/repo*
- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio? Para hacer un pull request podemos hacerlo desde github seleccionando el repositorio del que hicimos un fork previamente y la rama de nuestro repositorio ya "forkeado" y hacer el pull request desde la interfaz:



- ¿Cómo aceptar una solicitud de extracción? Para aceptar un pull request que nos hayan realizado, vamos al repositorio remoto en cuestión (en este caso en GitHub) vemos las solicitudes que tengamos y las aprobamos dependiendo de los cambios que veamos, analizando los cambios en código.



- ¿Qué es una etiqueta en Git? Una etiqueta en git es una “marca” que se deja para marcas puntos específicos del historial de desarrollo como importantes, se usa generalmente para marcar “versiones”.
- ¿Cómo crear una etiqueta en Git? Para crear una etiqueta se ejecuta el comando `git tag -a v1.4 -m “Detalle de la versión 1.4”`
- ¿Cómo enviar una etiqueta a GitHub? El comando `git push` no envía las etiquetas por defecto, para hacerlo debemos enviarlas de formar literal haciendo `git push origin v1.4` como si de una rama se tratase.
- ¿Qué es un historial de Git? El historial de Git son los cambios que hacemos de forma local en nuestro código y guardamos mediante un commit. Una vez que lo guardamos queda un registro único, un punto de control, para poder ver cómo estaba nuestro código en el momento que lo “commiteamos”.
- ¿Cómo ver el historial de Git? Para ver este historial debemos ejecutar el código `git log`
- ¿Cómo buscar en el historial de Git? Cuando ejecutamos el comando anterior nos muestra todos los commits en orden cronológico con fecha, hora y el detalle que hayamos puesto en el mensaje del commit. Para seleccionar un commit ver su detalle podemos ejecutar `git reflog` y nos mostrará en el visual los cambios que se hicieron en cada uno.

```

Marcelo@ELCOMANPC MINGW64 ~/Desktop/UTN/2025/Programación 1/2-Trabajo colaborativo (master)
$ git log
commit 92fe270049703fa6457bb948de15fd417b549e4a (HEAD -> master)
Merge: 39b08e4 ea5978c
Author: Marcelo Oviedo <oviedommarcelo@gmail.com>
Date: Wed Mar 26 14:53:23 2025 -0300

    Merge branch 'rama-prueba'

commit 39b08e42fc293797e2df6a890e230a6bb4724bb3
Author: Marcelo Oviedo <oviedommarcelo@gmail.com>
Date: Wed Mar 26 14:47:11 2025 -0300

    add comments

commit ea5978c09ae7415e255befdcc911a5c812ddd4dd (rama-prueba)
Author: Marcelo Oviedo <oviedommarcelo@gmail.com>
Date: Wed Mar 26 14:44:59 2025 -0300

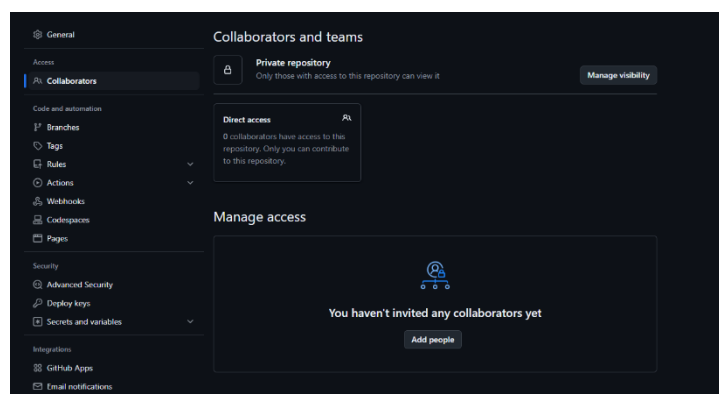
```

- ¿Cómo borrar el historial de Git? Para borrar todo el historial podríamos ir a la carpeta de nuestro repositorio y si tenemos en Windows disponible para ver los archivos ocultos borramos la carpeta `.git` y perderíamos todo. Ahora bien, también se puede ejecutar por la

consola ejecutando este comando `rm -rf .git`, que igualmente a fines práctico realiza esto mismo.

- ¿Qué es un repositorio privado en GitHub? Es un repositorio que no esta disponible de forma pública para que otros usuarios puedan verlo, colaborar o hacer un fork, deberíamos darle permisos explícitos a ese o esos usuarios.
- ¿Cómo crear un repositorio privado en GitHub? Cuando creamos nuestro repositorio debemos elegir la opción “Private”

- ¿Cómo invitar a alguien a un repositorio privado en GitHub? Para esto debemos ir a nuestro repositorio privado, ir a las settings o configuración y nos dirigimos a la opción Collaborators (colaboradores) y ahí podremos agregar a los usuarios que queremos invitar a colaborar.



- ¿Qué es un repositorio público en GitHub? Un repositorio público es un repo al cual cualquier persona puede ejecutar un pull request, un clone o un fork del proyecto para poder trabajar o colaborar sobre el mismo.
- ¿Cómo crear un repositorio público en GitHub? Dentro de nuestro usuario vamos a repositories, seleccionamos new y al generar tildamos la opción “public”

- ¿Cómo compartir un repositorio público en GitHub? Para compartirlo simplemente con el link del repositorio las personas ya están lista para poder colaborar con el proyecto.

2) Realizar la siguiente actividad:

Repositorio de la primera actividad: <https://github.com/OviedoMarcelo/python-test-git>

- 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:

Repositorio de ejercicio: <https://github.com/OviedoMarcelo/conflict-exercise>

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 de archivo README.md.

Paso 6: Resolver el conflicto

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

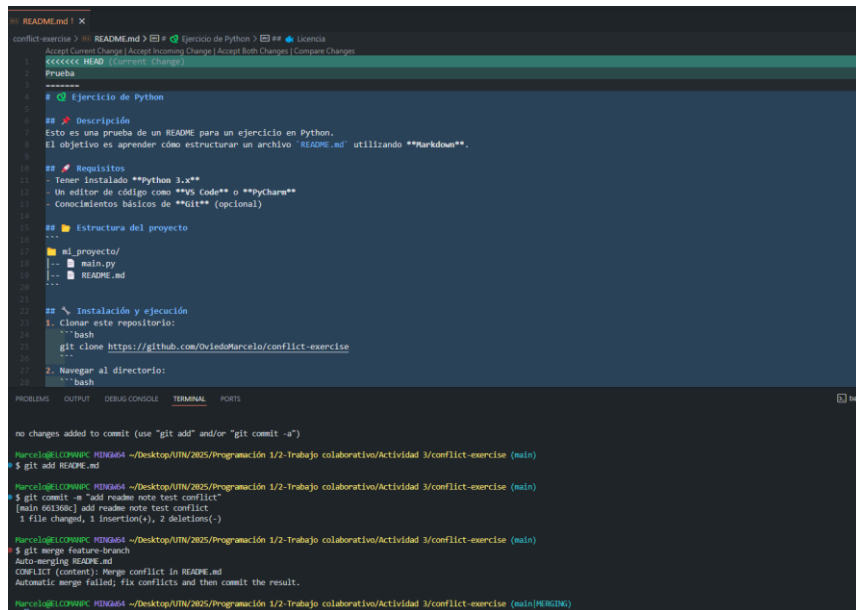
```
<<<<<<< HEAD
```

Este es un cambio en la main branch.

```
=====
```

Este es un cambio en la feature branch.

```
>>>>>>> feature-branch
```



```
conflict-exercise > README.md
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<< HEAD (current changes)
Prueba
-----
# Ejercicio de Python

## Descripción
Esto es una prueba de un README para un ejercicio en Python.
El objetivo es aprender cómo estructurar un archivo 'README.md' utilizando **Markdown**.

## Requisitos
- Tener instalado **Python 3.x**
- Un editor de código como **VS Code** o **PyCharm**
- Conocimientos básicos de **Git** (opcional)

## Estructura del proyecto
...
mi_proyecto/
├── main.py
└── README.md
...

## Instalación y ejecución
1. Clonar este repositorio:
```bash
git clone https://github.com/OvidioMarcelo/conflict-exercise
```
2. Navegar al directorio:
```bash
cd conflict-exercise
```

no changes added to commit (use "git add" and/or "git commit -a")
Narciso@GELCOMARPC: ~/Desktop/UTN/2025/Programación 1/2-Trabajo colaborativo/Actividad 3/conflict-exercise (main)
$ git add README.md
Narciso@GELCOMARPC: ~/Desktop/UTN/2025/Programación 1/2-Trabajo colaborativo/Actividad 3/conflict-exercise (main)
$ git commit -m "add readme note test conflict"
[main 661368c] add readme note test conflict
1 file changed, 1 insertion(+), 2 deletions(-)
Narciso@GELCOMARPC: ~/Desktop/UTN/2025/Programación 1/2-Trabajo colaborativo/Actividad 3/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.
Narciso@GELCOMARPC: ~/Desktop/UTN/2025/Programación 1/2-Trabajo colaborativo/Actividad 3/conflict-exercise (main|MERGING)
$
```

- 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.