

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?

Una comunidad donde podemos compartir nuestros repositorios en forma pública o privada. Seguir a otros usuarios, que ellos nos sigan, También puedes ver otros repositorios, podemos clonar o agregarlos en favoritos

- ¿Cómo crear un repositorio en GitHub?

Abrimos Git en modo consola , y empezamos a iniciar nuestro repositorio :

`git init` (inicializamos nuestro repositorio)

- ¿Cómo crear una rama en Git?

`git branch "(nombre de la rama que quieras agregar)"`

- ¿Cómo cambiar a una rama en Git?

`git checkout "(nombre de la rama a la que quieras cambiar)"`

- ¿Cómo fusionar ramas en Git?

`git checkout main` (nos paramos en la rama principal)

`git merge "(nombre de la rama que quieras fusionar) "`

- ¿Cómo crear un commit en Git?

`git add .` (agregamos los archivos al escenario de git)

`git commit -m "(nombre del mensaje que quieras iniciar) "`

- ¿Cómo enviar un commit a GitHub?

`git push -u origin main`

- ¿Qué es un repositorio remoto?

Son aquellos que se encuentran alojados en algún servidor externo y que pueden ser accedido desde cualquier lugar. Un ejemplo de repositorio remoto puede ser gitHub

- ¿Cómo agregar un repositorio remoto a Git?

Agregamos la dirección de nuestro repositorio remoto a nuestra consola (`git remote add origin <URL-del-repositorio>`)

`git push -u origin "(rama)"`

- ¿Cómo empujar cambios a un repositorio remoto?

`Git add .`

`Git commit -m "(agregamos el nombre al cambio que realizamos)"`

`Git push`

- ¿Cómo tirar de cambios de un repositorio remoto?

`Git pull origin "(nombre de la rama) "`

- ¿Qué es un fork de repositorio?

Fork es una copia de un repositorio , te permite hacer cambios libremente sin afectar el proyecto original.

- ¿Cómo crear un fork de un repositorio?

Nos dirigimos a la pagina del repositorio original al cual queremos clonar

Copiamos el url de ese repositorio

Nos dirigimos a donde dice “fork”

Creamos el “fork”

Una vez creado el “fork ” ya vamos a poder trabajar en el

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

Git checkout -b “(ponemos el nombre de la nueva rama)”

Git add .

Git commit -m “(nombre que se asigno cuando se creo la rama)”

Git push -u “(nombre asignado)”

Abrimos GitHub , ahí nos va a aparecer que se subió una nueva rama con cambios y te preguntara si quieres crear una pull request

Seleccionamos la base y la rama (donde hice los cambios) la que quiero comparar, y ahí le damos al botón verde “créate pull request”

Esperamos la revisión del grupo con el que te encuentras trabajando

- ¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción, tienes que ser el propietario del repositorio tener permiso de colaborador para poder fusionar los cambios

Apretar dentro del repositorio , donde se encuentra pull request y elegir la que sea desea revisar

Ahí mismo se te va a aparecer para revisar los cambios , ya que se muestran las diferencias entre las ramas

Ya estas listo para poder aceptar la pull request, GitHub te preguntara si las quieres fusionar la pull request

Merge commit

Rebase and merge

Squash and merge

Luego GitHub te va a preguntar si deseas eliminar la rama de pull request (delete Branch)

- ¿Qué es un etiqueta en Git?

Las etiquetas sirven para identificar un commit en particular en el repositorio. Se utilizan para versiones, lanzamientos o hechos clave del proyecto

- ¿Cómo crear una etiqueta en Git?

Git tag "(nombre que le quieras asignar a la etiqueta)"

- ¿Cómo enviar una etiqueta a GitHub?

Git push –tags

- ¿Qué es un historial de Git?

En el historial de git podemos ver todos los cambios que se han realizado en el repositorio incluyendo los cambios de los distintos commit que hemos realizado (autores, mensaje, cuando se hicieron)

- ¿Cómo ver el historial de Git?

Git log

- ¿Cómo buscar en el historial de Git?

Git log -n 2 (podemos ver los últimos dos commit realizados)

Git log "(nombre del archivo en específico que quieras ver)"

- ¿Cómo borrar el historial de Git?

Git reset –soft head~1 (elimina el ultimo commit)

Git reset --hard HEAD~<número-de-commits> (elimina varios commit)

git rebase -i <commit-id> (elimino un commit en específico)

- ¿Qué es un repositorio privado en GitHub?

Un repositorio privado es donde el acceso se encuentra en privado , solamente para personas que tenga acceso a el (miembros de un equipo o personas invitadas por el que creo el repositorio). Se usa mucho para proyectos confidenciales, comerciales o personales

- ¿Cómo crear un repositorio privado en GitHub?

Nos dirigimos a la parte donde se encuentra la palabra new (dentro de github)

Introducimos el nombre del repositorio que vamos a crear

Entre la opción de public o private lo ponemos en private y luego pulsamos el create repository

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

Vamos a settings, vamos a Collaborators y de ahí ponemos donde dice add people

Y ahí ponemos el nombre del usuario que queremos invitar a formar parte del repositorio

- ¿Qué es un repositorio público en GitHub?

Un repositorio publico es aquel que se crea en forma publica , es decir , que cualquiera puede tener acceso a el , clonarlo .Son ideales para proyectos de código abiertos

- ¿Cómo crear un repositorio público en GitHub?

Nos dirigimos a la parte donde se encuentra la palabra new (dentro de github)

Introducimos el nombre del repositorio que vamos a crear

Entre la opción de public o private lo ponemos en public y luego pulsamos el create repository

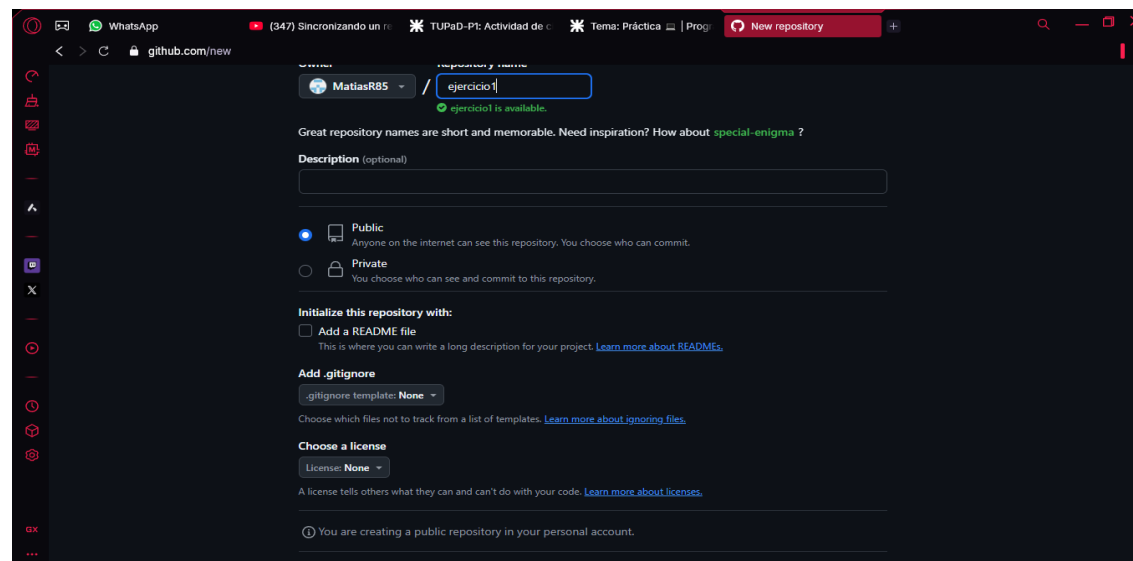
- ¿Cómo compartir un repositorio público en GitHub?

Vamos a nuestro repositorio creado, buscamos donde dice code (en verde) y copiamos el url de nuestro repositorio que se encuentra en forma publica

2) Realizar la siguiente actividad:

<https://github.com/MatiasR85/ejercicio1.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.

```
MINGW64:/c/Users/Mati/Desktop/Mi-archivo.txt

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/Mi-archivo.txt
$ git add .
fatal: not a git repository (or any of the parent directories): .git

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/Mi-archivo.txt
$ git init
Initialized empty Git repository in C:/Users/Mati/Desktop/Mi-archivo.txt/.git/

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/Mi-archivo.txt (master)
$ git add .

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/Mi-archivo.txt (master)
$ git commit -m "agregando mi archivo-txt"
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/Mi-archivo.txt (master)
$
```

- Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).

```
MINGW64:/c:/Users/Mati/Desktop/Mi-archivo.txt
$ git init
Reinitialized existing Git repository in C:/Users/Mati/Desktop/Mi-archivo.txt/.git/

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/Mi-archivo.txt (main)
$

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/Mi-archivo.txt (main)
$ git add .

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/Mi-archivo.txt (main)
$ git commit -m "agregando mi archivo.txt"
[main (root-commit) 7bf8982] agregando mi archivo.txt
1 file changed, 1 insertion(+)
create mode 100644 holaaa.txt

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/Mi-archivo.txt (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 224 bytes | 224.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
```

• Creando Branchs

- Crear una Branch
- Realizar cambios o agregar un archivo
- Subir la Branch

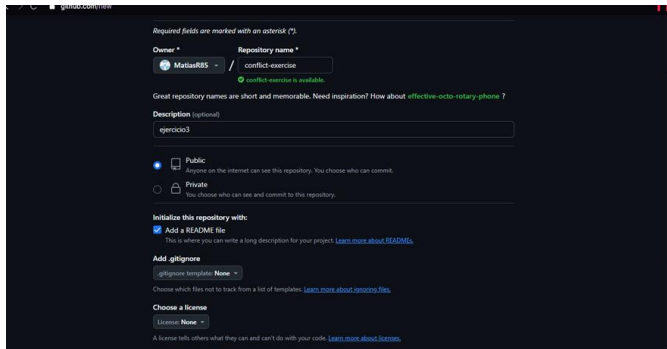
3) Realizar la siguiente actividad:

<https://github.com/MatiasR85/conflict-exercise.git>

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/Mati/Desktop/conflict-exercise
Mati@DESKTOP-8FORN39 MINGW64 ~/Desktop
$ git clone https://github.com/MatiasR85/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
Mati@DESKTOP-8FORN39 MINGW64 ~/Desktop
$ cd conflict-exercise
Mati@DESKTOP-8FORN39 MINGW64 ~/Desktop/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

- 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
```

```
MINGW64:/c:/Users/Mati/Desktop/conflict-exercise
1 file changed, 3 insertions(+), 1 deletion(-)

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/conflict-exercise (main)
$ git add README.md

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 373dbaa] Added a line in main branch
1 file changed, 1 insertion(+), 1 deletion(-)

Mati@DESKTOP-8F0RN39 MINGW64 ~/Desktop/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.
```

- 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
```

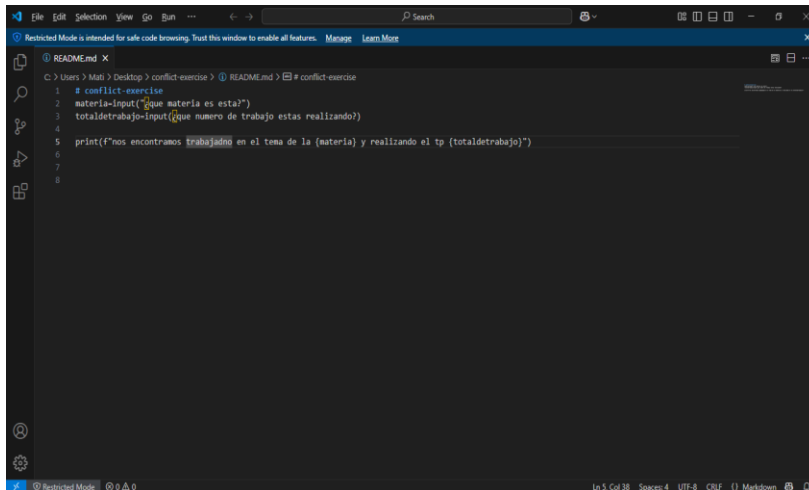
Este es un cambio en la main branch.

```
=====
```

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:



```
1 # conflict-exercise
2 materia-input("¿que materia es esta?")
3 totaldetrabajo-input("¿que numero de trabajo estas realizando?")
4
5 print(f"nos encontramos trabajando en el tema de la {materia} y realizando el tp {totaldetrabajo}")
6
7
8
```

`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.

