

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

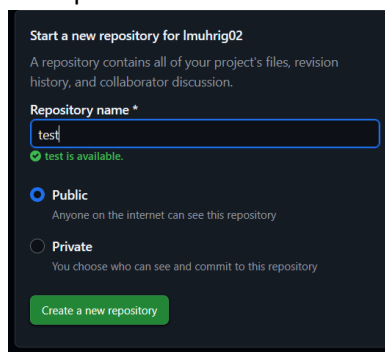
- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

- ¿Qué es GitHub?

GitHub es una Plataforma online donde alojar tu repositorio de Git. Se usa para colaborar con otros desarrolladores

- ¿Cómo crear un repositorio en GitHub?

Puedes hacer Fork (copiar) un repositorio de otra persona, o crear uno nuevo con esta opción:



Y luego hacer push de tu repositorio de GIT local o crear uno nuevo siguiendo las instrucciones que te da la plataforma.

- ¿Cómo crear una rama en Git?

Con el comando 'Git Branch nombre'

- ¿Cómo cambiar a una rama en Git?

Con el comando 'Git checkout nombreDeLaRama'

- ¿Cómo fusionar ramas en Git?

Posícionalte usando 'checkout' en la rama a donde harás la fusión, luego usa el comando 'Git merge nombreDeLaRama'

- ¿Cómo crear un commit en Git?

Primero haz 'Git add .' luego 'git commit -m "descripción del cambio"'

- ¿Cómo enviar un commit a GitHub?

Posiciónate en la rama correcta, luego haz 'git push origin master'

- ¿Qué es un repositorio remoto?

Es un repositorio alojado en un servidor online que permite la colaboración entre desarrolladoras en un mismo proyecto, sincronizando cambios

- ¿Cómo agregar un repositorio remoto a Git?

Haz 'git remote add origin url'

- ¿Cómo empujar cambios a un repositorio remoto?

Haz 'git push -u origin master' o 'git push' (Luego de la primera vez)

- ¿Cómo tirar de cambios de un repositorio remoto?

Haz 'git pull origin nombreDeLaRama' para descargar los cambios del repositorio y fusionarlos al repo local

- ¿Qué es un fork de repositorio?

Es una copia personal del repositorio que se almacena en tu cuenta de GitHub, permitiéndote editar el proyecto sin afectar al original.

- ¿Cómo crear un fork de un repositorio?

Ir al repositorio en GitHub, clicar el botón de Fork

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

Modifica tu fork o rama y haz commit, luego en GitHub ve a tu fork y usa el botón de “Pull request”. Selecciona la rama, agrega título y descripción, y clickea “créate pull request”
- ¿Cómo aceptar una solicitud de extracción?

Ve a la página del pull request, revisa los cambios, clickea en “Merge pull request”
- ¿Qué es una etiqueta en Git?

Es una referencia a un punto específico en la historia del repositorio. Marca versiones o hitos en el proyecto
- ¿Cómo crear una etiqueta en Git?

Haz `git tag -a v1.0 -m "Descripción de la etiqueta"`

-a es el nombre y -m el mensaje de descripción
- ¿Cómo enviar una etiqueta a GitHub?

Haz `git push origin v1.0` reemplaza v1.0 con tu tag
- ¿Qué es un historial de Git?

Es la secuencia de cambios realizados en el repositorio
- ¿Cómo ver el historial de Git?

Haz `git log`
- ¿Cómo buscar en el historial de Git?

Con `git log --grep="término de búsqueda"` o `git log -S "palabra clave"` o `git reflog`
- ¿Cómo borrar el historial de Git?

Con `git rebase -i --root`
- ¿Qué es un repositorio privado en GitHub?

Es un repo que solo puedes ver si estás autorizado
- ¿Cómo crear un repositorio privado en GitHub?

Cuando vas a GitHub y vas a New, cambia visibility a Private
- ¿Cómo invitar a alguien a un repositorio privado en GitHub?

En el repo, ve a settings > manage Access > invite a collaborator

- ¿Qué es un repositorio público en GitHub?

Es un repo accesible por cualquier persona

- ¿Cómo crear un repositorio público en GitHub?

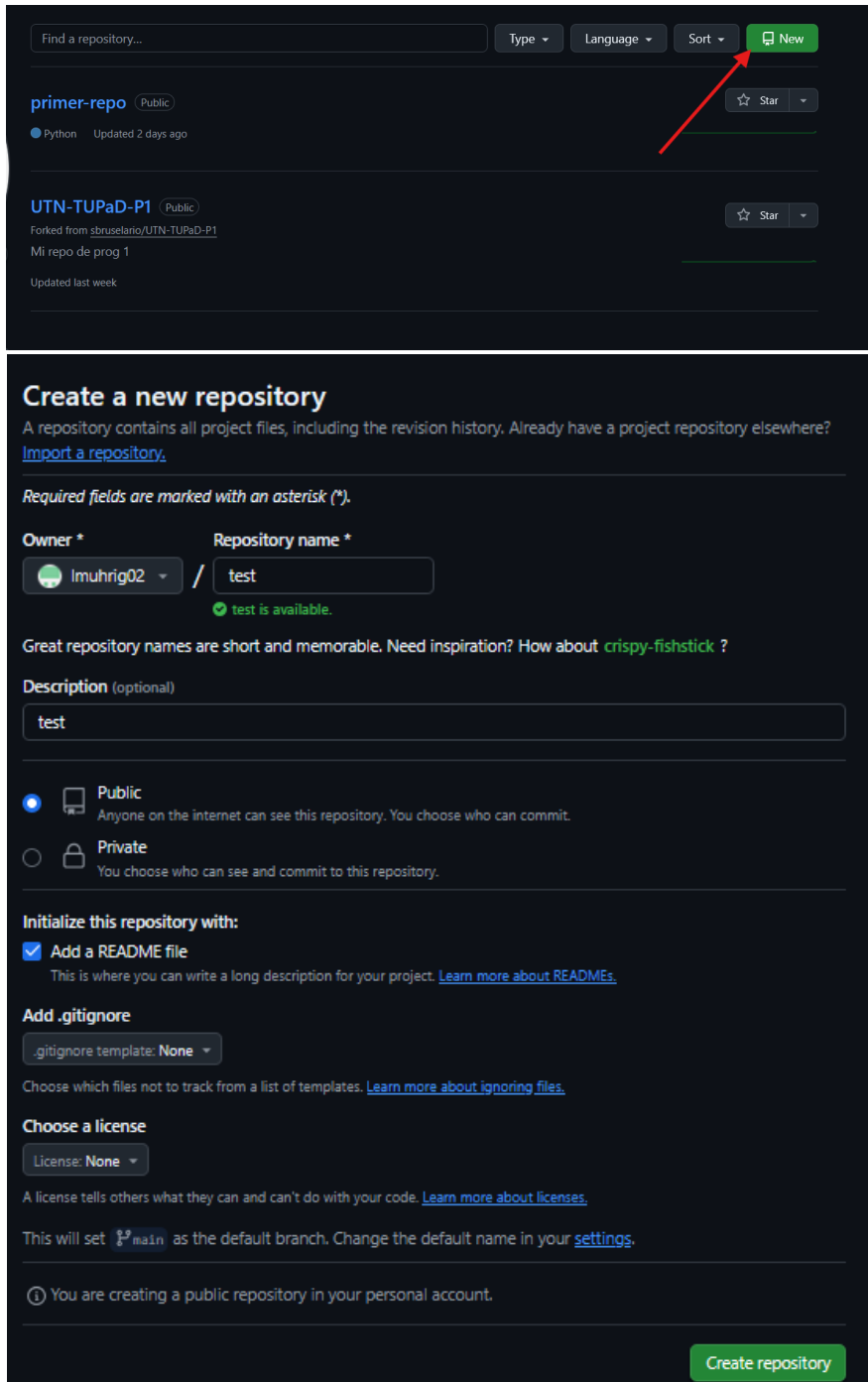
Cuando vas a New, asegúrate de poner visibility en Public

- ¿Cómo compartir un repositorio público en GitHub?

Copia la URL y compartela

2) Realizar la siguiente actividad:

- Crear un repositorio.
 - Dale un nombre al repositorio. ○ Elige el repositorio sea público.
 - Inicializa el repositorio con un archivo.



The image shows two screenshots from the GitHub website. The top screenshot shows the repository search and filter interface with a red arrow pointing to the 'New' button. The bottom screenshot shows the 'Create a new repository' form with the following details:

- Repository name:** test (with a green checkmark indicating it is available)
- Owner:** Imuhrig02
- Description:** test
- Visibility:** Public (selected)
- Initialize this repository with:** Add a README file (checked)
- Add .gitignore:** .gitignore template: None
- Choose a license:** License: None
- Footer:** You are creating a public repository in your personal account.

The 'Create repository' button is visible at the bottom right of the form.

Luego con los siguientes pasos me di cuenta que iba a necesitar primero crear el repositorio local con Git por lo que borre el repositorio de github ya que lo había incializado con un .txt

Cree uno nuevo sin inicializarlo con .txt

Y pushié mi nuevo repositorio local

Antes de esto, hice una nueva carpeta en mi desktop, hice CD hasta la carpeta con git bash

Hice git init

Y seguí las instrucciones de github para subirlo

- 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).

En VSCode cree un nuevo archivo .txt y lo guardé en mi repo local y luego pushie esto al github

Hice git add . y git commit -m "Agregando mi-archivo.txt"

Modifiqué el contenido de mi txt

Hice git add . y git commit

Ahora hice git push y vi que el txt en github se actualizó

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

Hice git checkout -b nuevaRama

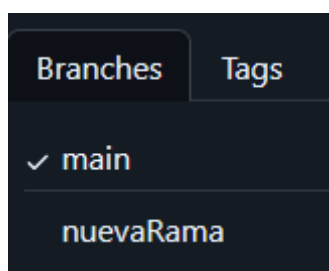
Agregue un nuevo archivo "nuevaRama.txt"

Hice git add nuevaRama.txt

Hice git comit -m "Agregando nuevaRama.txt"

Hice git push -u origin nuevaRama

Y verifico que aparece en github



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.

Siguiendo los pasos, a continuación una screenshot del git CMD y del conflicto en VS Code

```
C:\Users\Leo>git clone https://github.com/lmuhrig02/conflict-exercise
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.

C:\Users\Leo>cd conflict-exercise

C:\Users\Leo\conflict-exercise>git checkout -b feature-branch
Switched to a new branch 'feature-branch'

C:\Users\Leo\conflict-exercise>git add README.md

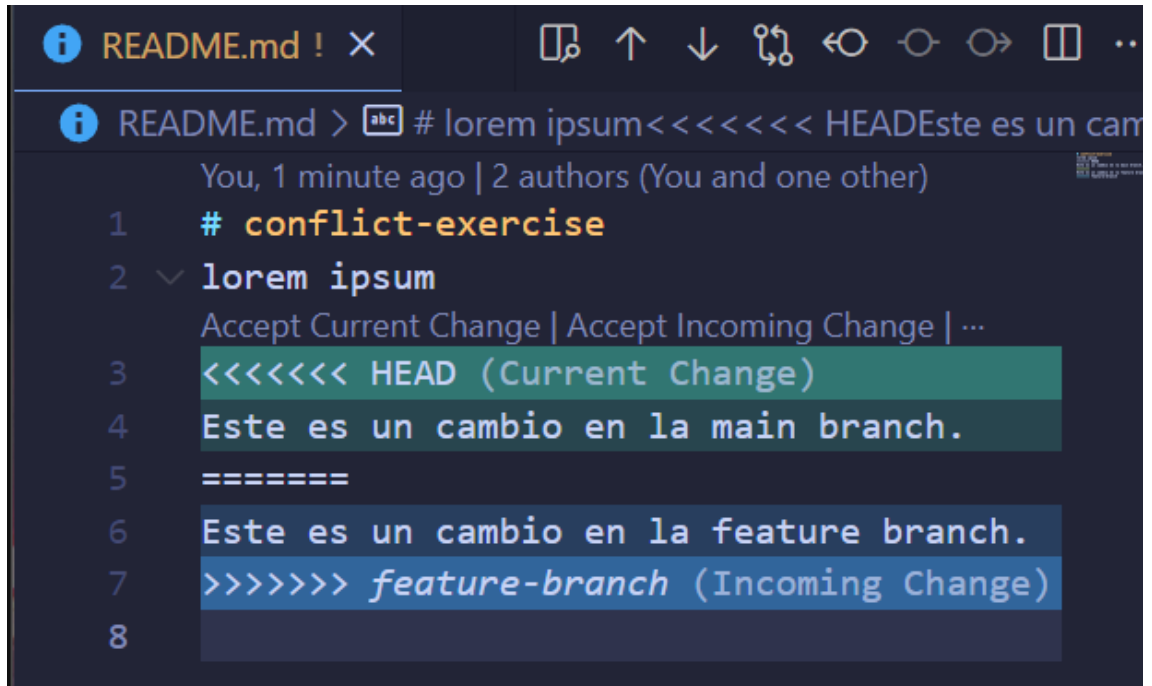
C:\Users\Leo\conflict-exercise>git commit -m "Added a line in feature-branch"
[feature-branch 917bffc] Added a line in feature-branch
1 file changed, 1 insertion(+)

C:\Users\Leo\conflict-exercise>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\Leo\conflict-exercise>git add README.md

C:\Users\Leo\conflict-exercise>git commit -m "Added a line in main branch"
[main 2f725ad] Added a line in main branch
1 file changed, 1 insertion(+)

C:\Users\Leo\conflict-exercise>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.
```

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 (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

C:\Users\Leo\conflict-exercise>git add README.md

C:\Users\Leo\conflict-exercise>git commit -m "Resolved merge conflict"
[main d8c6a3f] Resolved merge conflict

C:\Users\Leo\conflict-exercise>git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 934 bytes | 934.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/lmuhrig02/conflict-exercise
028eeae..d8c6a3f  main -> main

C:\Users\Leo\conflict-exercise>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/lmuhrig02/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/lmuhrig02/conflict-exercise
* [new branch]   feature-branch -> feature-branch

C:\Users\Leo\conflict-exercise>git add README.md

C:\Users\Leo\conflict-exercise>git commit -m "fixed typo"
[main a8d0603] fixed typo
1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\Leo\conflict-exercise>git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 268 bytes | 268.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/lmuhrig02/conflict-exercise
d8c6a3f..a8d0603  main -> main
```

