TRABAJO PRACTICO COLABORATIVO
MAURO PONCE 42266883
COMISION 19

TRABAJO PRACTICO COLABORATIVO

ACTIVIDADES

Ejercicio 1

¿Qué es GitHub?

GitHub es una plataforma para alojar y colaborar en proyectos de código usando Git, un sistema de control de versiones. Digamos un lugar en la nube donde guardas el código, lo compartes con otros y trabajas en equipo, con historial de cambios y todo organizado.

¿Cómo crear un repositorio en GitHub?

1. Iniciar sesión en GitHub.
2. Haz clic en el botón "+" en la esquina superior derecha y selecciona "New repository".
3. Darle un nombre, elegir si será público o privado, y agregar una descripción o un archivo README.
4. Clic en "Create repository".

¿Cómo crear una rama en Git?

En la terminal de Visual Studio Code o de la misma terminal de git, usar el siguiente comando:

git branch nombre-rama

Esto crea una rama nueva llamada "nombre-rama"

¿Cómo cambiar a una rama en Git?

En la terminal, usar el comando:
git checkout nombre-rama

¿Cómo fusionar ramas en Git?

1. Cambiar a la rama principal (ejemplo: git checkout main).
2. Fusionar la otra rama con:
   git merge nombre-rama
   Ejemplo: git merge otra-rama. Si hay conflictos, Git te avisa y los resuelves manualmente.

¿Cómo crear un commit en Git?

1. Agregar los archivos que queramos incluir con:
   git add. (todos) o git add archivo.txt (uno en específico).
2. Crea el commit con:
   git commit -m "Mensaje descriptivo"
   Ejemplo: git commit -m "Agrego función de login".

¿Cómo enviar un commit a GitHub?

Usar en la terminal:
git push origin nombre-rama
Ejemplo: git push origin master o main. Esto sube los cambios al repositorio remoto en Git.

¿Qué es un repositorio remoto?

Es una versión de tu proyecto alojada en un servidor (como GitHub), no en la computadora. Sirve para compartir y respaldar el trabajo.

¿Cómo agregar un repositorio remoto a Git?

Con este comando: git remote add origin URL-del-repositorio
La URL se encuentras en GitHub al clonar o crear el repo. Ejemplo: git remote add origin https://github.com/tu-usuario/tu-repo.git.

¿Cómo empujar cambios a un repositorio remoto?

Con:
git push origin nombre-rama
Ejemplo: git push origin main. Esto "empuja" tus commits al remoto.

¿Cómo tirar de cambios de un repositorio remoto?

Usa:
git pull origin nombre-rama
Ejemplo: git pull origin main. Esto trae los cambios del remoto a la máquina local.

¿Qué es un fork de repositorio?

Un fork es una copia de un repositorio de otro usuario en tu cuenta de GitHub. Sirve para trabajar en él sin afectar el original.

¿Cómo crear un fork de un repositorio?

1. Vas al repositorio en GitHub que quieres copiar.
2. Haces clic en el botón "Fork" arriba a la derecha.
3. Esperasaun momento, y aparecerá en tu cuenta como tu propio repositorio.

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

1. Se hace un fork y clona tu copia: git clone URL.
2. Creas una rama, haces cambios, y se suben: git push origin tu-rama.
3. En GitHub, vas a tu fork, haz clic en "Pull request", selecciona las ramas y envías la solicitud.

¿Cómo aceptar una solicitud de extracción?

1. Vas al repositorio original en GitHub.
2. En la pestaña "Pull requests", selecciona la solicitud.
3. Revisa los cambios y haz clic en "Merge pull request" si todo está bien, luego "Confirm merge".

¿Qué es una etiqueta en Git?

Es como un marcador que pones en un commit específico, generalmente para señalar versiones (ejemplo: v1.0).

¿Cómo crear una etiqueta en Git?

Usa:
git tag nombre-etiqueta
Ejemplo: git tag v1.0. Se pone en el commit actual.

¿Cómo enviar una etiqueta a GitHub?

Con:
git push origin nombre-etiqueta
Ejemplo: git push origin v1.0.

¿Qué es un historial de Git?

Es el registro de todos los commits que has hecho, con detalles como autor, fecha y mensaje.

¿Cómo ver el historial de Git?

Usar:
git log
Para algo más simple:
git log --oneline (solo muestra una línea por commit).

¿Cómo buscar en el historial de Git?

Con:
git log --grep="palabra"
Ejemplo: git log --grep="login" buscar commits con "login" en el mensaje.

¿Cómo borrar el historial de Git?

Podes reiniciar todo con:
git reset --hard HEAD (borra cambios no confirmados) o reescribir el historial con git rebase.

¿Qué es un repositorio privado en GitHub?

Es un repositorio que solo vos y las personas que invites pueden ver o editar.

¿Cómo crear un repositorio privado en GitHub?

Igual que uno público, pero al crearlo, marca la opción "Private" en vez de "Public" antes de darle a "Create repository".

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

1. Vas a tu repositorio, haz clic en "Settings".
2. En "Collaborators", buscas al usuario por nombre o email y solo queda agregarlo.

¿Qué es un repositorio público en GitHub?

Es uno que cualquiera puede ver y clonar, aunque solo los colaboradores pueden editarlo.

¿Cómo crear un repositorio público en GitHub?

Como el privado, pero seleccionas "Public" al crearlo. Simple.

¿Cómo compartir un repositorio público en GitHub?

Solo pasa la URL del repositorio (ejemplo: https://github.com/tu-usuario/tu-repo).

PRACTICA

```
PC@DESKTOP-H9N96U4 MINGW64 ~
$ cd mi-primer-repositorio

PC@DESKTOP-H9N96U4 MINGW64 ~/mi-primer-repositorio (main)
$ git branch nueva-rama

PC@DESKTOP-H9N96U4 MINGW64 ~/mi-primer-repositorio (main)
$ git checkout nueva-rama
Switched to branch 'nueva-rama'

PC@DESKTOP-H9N96U4 MINGW64 ~/mi-primer-repositorio (nueva-rama)
$ git branch
  main
* nueva-rama

PC@DESKTOP-H9N96U4 MINGW64 ~/mi-primer-repositorio (nueva-rama)
$ echo "Este es un archivo en la nueva rama" > otro-archivo.txt

PC@DESKTOP-H9N96U4 MINGW64 ~/mi-primer-repositorio (nueva-rama)
$ git add .
warning: in the working copy of 'otro-archivo.txt', LF will be replaced by CRLF the next time Git touc
hes it

PC@DESKTOP-H9N96U4 MINGW64 ~/mi-primer-repositorio (nueva-rama)
$ git commit -m "Agregando otro-archivo.txt en nueva-rama"
[nueva-rama e45ca8e] Agregando otro-archivo.txt en nueva-rama
 1 file changed, 1 insertion(+)
 create mode 100644 otro-archivo.txt

PC@DESKTOP-H9N96U4 MINGW64 ~/mi-primer-repositorio (nueva-rama)
$ git push origin nueva-rama
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 1.43 KiB | 1.43 MiB/s, done.
Total 9 (delta 1), reused 3 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'nueva-rama' on GitHub by visiting:
remote:      https://github.com/MauroProgram/mi-primer-repositorio/pull/new/nueva-rama
remote:
To https://github.com/MauroProgram/mi-primer-repositorio.git
 * [new branch]      nueva-rama -> nueva-rama

PC@DESKTOP-H9N96U4 MINGW64 ~/mi-primer-repositorio (nueva-rama)
$
```

```
PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (feature-branch)
$ code README.md

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (feature-branch)
$ git branch
  feature-brach
* feature-branch
  main

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (feature-branch)
$ git add README.md

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (feature-branch)
$ git commit -m"Added a line in feature-branch"
[feature-branch 093b520] Added a line in feature-branch
 1 file changed, 3 insertions(+), 1 deletion(-)

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main)
$ git add README.md

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main)
$ git commit -m "Added a line in a main branch"
[main bc2ac38] Added a line in a main branch
 1 file changed, 1 insertion(+)

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main)
$ git branch
  feature-brach
  feature-branch
* main

PC@DESKTOP-H9N96U4 MINGW64 ~/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.

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main|MERGING)
$ git add README.md

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main 64e4175] Resolved merge conflict

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main)
$ git push origin main
```

https://github.com/MauroProgram/conflict-excercise.git

Informe Detallado del Ejercicio de Resolución de Conflictos en Git

Propósito del Ejercicio:

El objetivo principal de esta actividad consistió en simular y resolver un conflicto de fusión dentro de un entorno de control de versiones Git. Esta habilidad se considera fundamental para el desarrollo colaborativo de software, donde múltiples contribuyentes pueden modificar el mismo código simultáneamente.

Desarrollo del Ejercicio:

1. **Creación del Repositorio Remoto:** Se procedió a la creación de un nuevo repositorio en la plataforma GitHub, denominado "conflict-exercise". Este repositorio fue inicializado con un archivo README.md básico.
2. **Clonación del Repositorio Local:** Utilizando el comando git clone, se realizó una copia del repositorio remoto en el sistema local, permitiendo así trabajar directamente sobre el código.
3. **Creación de la Rama feature-branch:** Se generó una nueva rama de desarrollo, denominada feature-branch, con el fin de simular el trabajo en una funcionalidad separada.
4. **Modificación del Archivo README.md en feature-branch:** Se realizó una modificación al archivo README.md dentro de la rama feature-branch, añadiendo una línea de texto. Los cambios fueron confirmados mediante un git commit.
5. **Retorno a la Rama Principal main:** Se volvió a la rama principal main mediante el comando git checkout main.
6. **Modificación del Archivo README.md en main:** Se efectuó una modificación diferente al mismo archivo README.md dentro de la rama main, y los cambios fueron confirmados con un git commit.
7. **Intento de Fusión y Conflicto:** Se intentó fusionar los cambios de feature-branch en main utilizando git merge feature-branch. Esto resultó en un conflicto, ya que ambas ramas habían modificado la misma línea del archivo README.md.
8. **Resolución Manual del Conflicto:** Se abrió el archivo README.md en un editor de texto y se procedió a resolver el conflicto manualmente, decidiendo qué cambios conservar. La resolución fue confirmada con un git commit.
9. **Envío de Cambios al Repositorio Remoto:** Se enviaron los cambios de la rama main al repositorio remoto en GitHub mediante git push origin main.
10. **Verificación en GitHub:** Se verificó que los cambios se reflejaran correctamente en el repositorio remoto a través de la interfaz web de GitHub.

Análisis de Errores y Aprendizajes:

Durante el desarrollo del ejercicio, se presentaron algunos errores que requirieron atención:

- **Error de Repositorio No Encontrado:** Inicialmente, se experimentó un error que indicaba que el repositorio remoto no se encontraba. Esto se debió a un error en la URL del repositorio.
- **Error de Rama "master" Inexistente:** Se produjo un error al intentar hacer push a una rama "master" inexistente, ya que la rama principal del repositorio era "main".
- **Error de push Rechazado:** Se experimentaron rechazos de push debido a que el repositorio remoto contenía cambios que no estaban presentes localmente.
- **Error de Repositorio Remoto Existente:** Se intentó agregar un repositorio remoto con el nombre "origin" cuando ya existía uno configurado, lo que generó un error.

Lecciones Aprendidas:

- La importancia de la precisión en los nombres de repositorios y ramas.
- La necesidad de comprender e interpretar los mensajes de error de Git.
- La práctica de utilizar git pull antes de git push para evitar conflictos.
- La verificación de la URL del repositorio remoto antes de realizar operaciones.
- Aprender a modificar la URL de un repositorio remoto ya existente.

```
remote: Repository not found.
fatal: repository 'https://github.com/MauroProgram/conflict-exercise.git/' not found

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main)
$ git push origin master
error: src refspec master does not match any
error: failed to push some refs to 'https://github.com/MauroProgram/conflict-exercise.g

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main)
$ git remote -v
origin  https://github.com/MauroProgram/conflict-exercise.git (fetch)
origin  https://github.com/MauroProgram/conflict-exercise.git (push)

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main)
$ git remote set-url origin https://github.com/MauroProgram/conflict-excercise.git

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main)
$ git push origin main
To https://github.com/MauroProgram/conflict-excercise.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/MauroProgram/conflict-excercise.
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main)
$ git remote add origin https://github.com/MauroProgram/conflict-excercise.git
error: remote origin already exists.

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main)
$ git push origin main
To https://github.com/MauroProgram/conflict-excercise.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/MauroProgram/conflict-excercise.
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

PC@DESKTOP-H9N96U4 MINGW64 ~/conflict-exercise (main)
$ git pull origin main
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)
Unpacking objects: 100% (3/3), 910 bytes | 101.00 KiB/s, done.
From https://github.com/MauroProgram/conflict-excercise
 * branch            main       -> FETCH_HEAD
```