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.

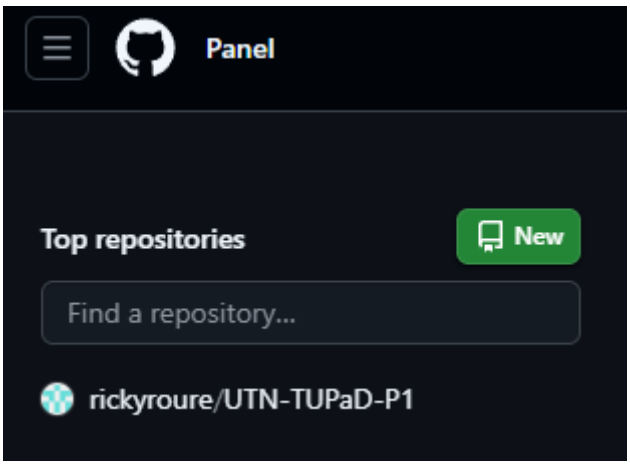
Actividades

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

- **¿Qué es GitHub?**
  GitHub es una plataforma para almacenar, gestionar y colaborar en proyectos de código fuente de manera remota.

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

  En la página principal, busca un botón verde que dice "New" o "Nuevo" en la esquina superior izquierda, cerca de tu foto de perfil.



Configura tu repositorio:

Nombre del repositorio: Elige un nombre único para tu repositorio. Este será el identificador de tu proyecto.

Descripción (opcional): Puedes añadir una breve descripción de lo que trata tu repositorio.

Visibilidad: Decide si quieres que el repositorio sea público (cualquiera puede verlo) o privado (solo tú y las personas que invites pueden verlo).

Iniciar con README (opcional): Si seleccionas esta opción, GitHub creará un archivo README donde puedes escribir una descripción más detallada de tu proyecto.

Crear el repositorio:

Haz clic en "Create repository" para finalizar la creación de tu repositorio.

- **¿Cómo crear una rama en Git?**

  Para crear una nueva rama en git usamos el comando: git Branch nombreDeLaRama

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

  Para cambiar a otra rama (tomando como ejemplo la rama creada en la pregunta anterior) usamos el comando: git checkout nombreDeLaRama

- **¿Cómo fusionar ramas en Git?**

  Es recomendable posicionarnos sobre la rama principal, (también llamada main o master) a la cual le vamos a agregar el contenido de la otra rama y colocamos el comando: git merge nombreDeLaRama

- **¿Cómo crear un commit en Git?**

  Para sacarle una “foto” al proyecto, usamos el comando: git commit -m “Reseña del avance del proyecto”

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

  Para "pushear" los cambios a GitHub debemos usar el comando: git push -u origin main

- **¿Qué es un repositorio remoto?**

  Es un repositorio que está guardado en línea, como en **GitHub**. Es el lugar donde puedes subir tu código para compartirlo con otros o guardarlo como copia de seguridad.

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

  Para agregar un repositorio demoto a git usamos el comando: git remote add origin [URL-del-repositorio]

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

  Git push -u origin master

  Git push (luego de la primera vez)

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

  Git pull origin master

  Git pull (si usamos -u en el push)

- **¿Qué es un fork de repositorio?**

  "Forkear" un repositorio es crear una copia de un repo X en nuestra propia cuenta. Es útil si quieres hacer cambios en un proyecto sin afectar el original.

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

  Para forkear un repositorio, localizamos el mismo, hacemos click en "Fork" (generalmente en la parte superior derecha). Esto creará una copia de ese repositorio en tu cuenta de GitH

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

  Después de hacer cambios en tu fork (copia) de un repositorio, puedes proponer esos cambios al repositorio original. Vas a tu fork en GitHub, y le das a **"New Pull Request"**. Luego, comparas tu fork con el original y envías la solicitud.

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

  Si eres el dueño del repositorio original, puedes ver la solicitud de extracción y aceptarla. Simplemente le das a **"Merge"** en GitHub para integrar los cambios.

- **¿Qué es un etiqueta en Git?**

  Una **etiqueta** (o **tag**) es como un marcador que pones en un punto específico del historial de tu proyecto, generalmente para marcar versiones importantes (por ejemplo, **v1.0**).

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

  Para crear una etiqueta, usas el siguiente comando: git tag -a v1.0 -m "Primera versión"

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

  Para enviar la etiqueta a GitHub, usas: git push origin v1.0

- **¿Qué es un historial de Git?**

  El **historial de Git** es una lista de todos los cambios que se han hecho en tu proyecto a lo largo del tiempo, con información sobre cuándo se hicieron y quién los hizo.

- **¿Cómo ver el historial de Git?**

  Usando el comando: git log te mostrará todos los cambios guardados con **commit**.

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

  Si quieres buscar un cambio específico en el historial, puedes usar: git log --grep="mensaje"

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

  git reset --hard [commit-id]

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

  Un **repositorio privado** es un repositorio donde solo las personas que tú invites pueden ver y modificar el código. No es visible para todos.

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

  Cuando creas un repositorio en GitHub, puedes elegir la opción **"Private"** para que sea privado.

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

  Para invitar a alguien, vas a la página de tu repositorio en GitHub, haces clic en **"Settings"** > **"Manage access"** y luego invitas a las personas por su nombre de usuario.

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

  Un **repositorio público** es un repositorio que cualquier persona puede ver y usar. No necesita invitación, y es accesible a todos en GitHub.

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

  Cuando creas un repositorio en GitHub, eliges la opción **"Public"** para que sea público.

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

  Solo necesitas copiar la URL del repositorio (por ejemplo, https://github.com/rickyroure/mi-primer-repo.git) y compartirla con quien quieras.

2) Realizar la siguiente actividad:

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

```
$ git clone https://github.com/rickyroure/mi-primer-repo.git
Cloning into 'mi-primer-repo'...
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.
```

- 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).

```
$ git add .

54351@DESKTOP-EH1MKQA MINGW64 ~/ricky/programacion/Programacior
 (main)
$ git commit -m "agregado"
[main 47d1ec1] agregado
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Nuevo documento de texto.txt

54351@DESKTOP-EH1MKQA MINGW64 ~/ricky/programacion/Programacior
 (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 294 bytes | 294.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/rickyroure/mi-primer-repo.git
   bbd6c1b..47d1ec1  main -> main
```

- Creando Branchs

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

```
$ git checkout -b nuevaRama
Switched to a new branch 'nuevaRama'

54351@DESKTOP-EH1MKQA MINGW64 ~/ricky/pro
(nuevaRama)
```

```
54351@DESKTOP-EH1MKQA MINGW64 ~/ricky/programacion/Programac
(nuevaRama)
$ git commit -m "Agregando modificacion desde nuevaRama"
[nuevaRama e7f8623] Agregando modificacion desde nuevaRama
 2 files changed, 1 insertion(+)
 delete mode 100644 Nuevo documento de texto.txt
 create mode 100644 texto.txt
```

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.

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:

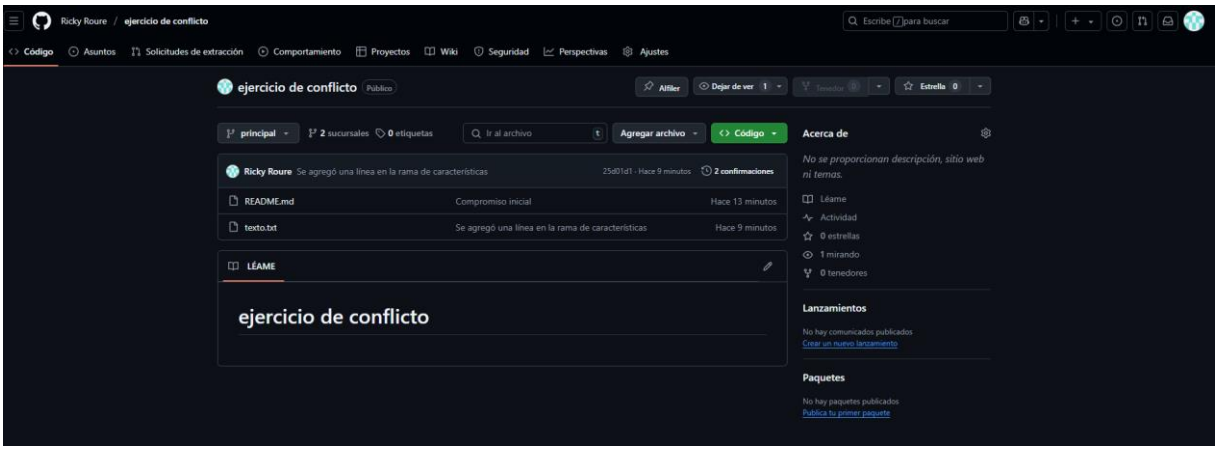
git push origin main

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.



URL de los repos:
https://github.com/rickyroure/mi-primer-repo
https://github.com/rickyroure/conflict-exercise