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?

Es una plataforma para guardar código (repositorio) que facilita trabajar en proyectos con otras personas usando Git.

¿Cómo crear un repositorio en GitHub?

Entrás a tu cuenta, tocás “New repository”, le ponés un nombre y listo.

¿Cómo crear una rama en Git?

Con el comando git branch nombre-de-la-rama.

¿Cómo cambiar a una rama en Git?

Usás git checkout nombre-de-la-rama.

¿Cómo fusionar ramas en Git?

Primero te pasás a la rama principal (main o master) y después hacés git merge nombre-de-la-rama.

¿Cómo crear un commit en Git?

Primero agregás los archivos con git add, después hacés git commit -m "tu mensaje".

¿Cómo enviar un commit a GitHub?

Con git push origin nombre-de-la-rama.

¿Qué es un repositorio remoto?

Es una copia del repo en la nube, por ejemplo en GitHub.

¿Cómo agregar un repositorio remoto a Git?

Usás git remote add origin URL-del-repo.

¿Cómo empujar cambios a un repositorio remoto?

Con git push origin nombre-de-la-rama.

¿Cómo tirar de cambios de un repositorio remoto?

Hacés git pull origin nombre-de-la-rama.

¿Qué es un fork de repositorio?

Es una copia de un repo de otra persona para que puedas realizar modificaciones sin afectar el original.

¿Cómo crear un fork de un repositorio?

Entrás al repo en GitHub y tocás el botón “Fork”.

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

Desde tu fork, hacés clic en “Pull Request” y seguís los pasos para proponer tus cambios.

¿Cómo aceptar una solicitud de extracción?

El dueño del repo revisa el pull request y, si acepta los cambios, lo mergea.

¿Qué es una etiqueta en Git?

Es como ponerle un marcador a un punto importante del proyecto, como una versión.

¿Cómo crear una etiqueta en Git?

Con git tag nombre-de-la-etiqueta.

¿Cómo enviar una etiqueta a GitHub?

Con git push origin nombre-de-la-etiqueta.

¿Qué es un historial de Git?

Es el registro de todos los cambios que se hicieron en el proyecto.

¿Cómo ver el historial de Git?

Con git log

¿Cómo buscar en el historial de Git?

Úsando git log --grep="palabra" para encontrar algo específico.

¿Cómo borrar el historial de Git?

No es algo común ni recomendable, pero se puede hacer con comandos avanzados como rebase o filter-branch.

¿Qué es un repositorio privado en GitHub?

Un repo que solo pueden ver vos y a quien invites.

¿Cómo crear un repositorio privado en GitHub?

Cuando lo creás, marcás la opción “Private”.

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

Vas a “Settings” > “Collaborators” y agregás su usuario.

¿Qué es un repositorio público en GitHub?

Es un repo que puede ver cualquiera.

¿Cómo crear un repositorio público en GitHub?

Al crearlo, dejás marcada la opción “Public”.

¿Cómo compartir un repositorio público en GitHub?

Le pasás la URL del repo a quien quieras.

2) Realizar la siguiente actividad:

https://github.com/ricapardo/TP_GIT_GITHUB.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:

https://github.com/ricapardo/TP_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
```

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.