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 de desarrollo colaborativo,para alojar, revisar y colaborar en proyectos..

• ¿Cómo crear un repositorio en GitHub?

En la plataforma de GitHub seleccionaremos crear un nuevo repositorio,luego de esto procedemos a configurarlo,le añadimos un nombre y una breve descripción y luego simplemente aceptamos todo.

• ¿Cómo crear una rama en Git?

Con el siguiente comando podremos crear una rama en Git (git branch nombre-de-la-rama)

• ¿Cómo cambiar a una rama en Git?

Lo primero que debemos hacer es visualizar todas las ramas locales con (git branch)
Luego de esto con (git checkout)y el nombre de la rama nos dirigimos hacia la rama seleccionada

• ¿Cómo fusionar ramas en Git?

Para fusionar las ramas debemos primero asegurarnos en que rama estamos y luego con un (git merge feature)fusionamos ambas ramas y confirmamos los cambios con un (git push origin main)

• ¿Cómo crear un commit en Git?

Para iniciar realizamos un status (git status),luego usamos un (git add .) y finalmente creamos un commit con un (git commit -m "Mensaje que querramos")

• ¿Cómo enviar un commit a GitHub?

Luego de crear nuestro commit como explicamos anteriormente y verificar nuestra rama actual deberemos hacer un push (git push origin nombre_rama)

• ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión de tus proyectos alojados en la nube (en nuestro caso GitHub)

• ¿Cómo agregar un repositorio remoto a Git?

Copia la URL de tu repo desde GitHub
Ejecuta en tu Cmd (git remote add origin URL .git)
Realiza un push (git push -u origin main)

• ¿Cómo empujar cambios a un repositorio remoto?

Realizar los cambios (git add .)
Crear un commit (git commit -m “descripción de los cambios”)
Subimos los cambios (git push origin nombre_rama)

• ¿Cómo tirar de cambios de un repositorio remoto?

Para obtener los cambios de un repositorio remoto debemos:
Primero realizar un checkout (git checkout nombre_rama)
Realizar un pull para obtener los cambios(git pull origin nombre_rama)

• ¿Qué es un fork de repositorio?

Un fork es una copia personal de un repositorio de GitHub que te permite modificar el código sin afectar el proyecto original,experimentar libremente en tu espacio y enviar mejoras del proyecto original.

• ¿Cómo crear un fork de un repositorio?

Para crear un fork debemos ir al repositorio original y hacer clic en “Fork” arriba a la derecha y se creara una réplica en tu cuenta que luego deberás clonar en tu equipo

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

Primero debemos crear un Fork del repositorio con el que trabajaremos
Lo clonamos en nuestro equipo

(git clone https://github.com/Ejemplo/repo.git)
Creamos una rama para los cambios(git checkout -b mi-feature)
Modificamos lo que debamos
(git add .
git commit -m "Descripción clara de los cambios"
git push origin mi-feature)
Luego de esto vamos a GitHub,entramos al fork y hacemos click en “Compare y Pull Request) y lo creamos
• ¿Cómo aceptar una solicitud de extracción?

Para iniciar revisamos los cambios
Luego usamos un “Merge pull request”
Y para confirmar totalmente agregamos un mensaje final y confirmamos con “Confirm Merge”

• ¿Qué es una etiqueta en Git?

En Git una etiqueta es una referencia a un punto específico en el historial de commits,usado principalmente para marcar versiones importantes

• ¿Cómo crear una etiqueta en Git?

Asegurarnos de estar en el commit correcto
(git checkout main
git log --oneline)
Crear una etiqueta (git tag v1.0.0)
Verificamos la etiqueta creada
(git tag
git show v1.0.0)
Subimos la etiqueta al repositorio remoto
(git push origin v1.0.0)

• ¿Cómo enviar una etiqueta a GitHub?

Crear una etiqueta local como lo explicamos anteriormente
La subimos a GitHub(git push origin v1.0.0)para una etiqueta específica(git push origin --tags)para todas las etiquetas locales

• ¿Qué es un historial de Git?

El historia de Git es el registro completo de los cambios de un repositorio,en el que podemos ver quien,cuando y qué cambios se realizaron

• ¿Cómo ver el historial de Git?

Para ver el historial de Git necesitamos usar el comando (git log)

• ¿Cómo buscar en el historial de Git?

Para buscar en el historial de Git podemos utilizar
palabras claves(git log --grep="palabra clave")
quien lo realizó(git log --author="nombre|email")
fechas(git log --since="2024-01-01" --until="2024-03-31")
archivos específicos(git log -- archivo.js)
contenido(git log -S "texto buscado")
combinar distintos métodos(git log --author="Juan" --grep="bug" --since="1 month ago" --pretty=format:"%h - %an, %ar : %s")
buscar commits que tocaron cierta línea(git blame archivo.txt)

• ¿Cómo borrar el historial de Git?

Si los archivos no se compartieron
(git reset --soft HEAD~2 # Borra los últimos N commits pero mantiene los cambios en staging

git reset --hard HEAD~2 # Borra completamente los últimos N commits (incluyendo los cambios))

• ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un proyecto cuyo código es solo visible para los usuarios a los que se les ha concedido permisos explícitamente

• ¿Cómo crear un repositorio privado en GitHub?

En GitHub hacemos click en “New Repository” y lo configuramos a nuestras necesidades,modificamos el nombre y en el apartado de visibilidad seleccionamos “Private”para hacerlo privado

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

Para invitar a alguien a nuestro repositorio privado debemos ingresar al apartado “Settings”-”Collaborators”-”Add people” y luego ingresamos el nombre de usuario o mail del colaborador

• ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un proyecto de código abierto que cualquier persona puede ver,clonar y contribuir

• ¿Cómo crear un repositorio público en GitHub?

Para crear un repo público debemos
Crear un repositorio “New repository”
Le ponemos un nombre y en la opción de visibilidad seleccionamos “Public” que es la opción predeterminada.

• ¿Cómo compartir un repositorio público en GitHub?

Para compartir tu repositorio público simplemente debemos copiar el URL del repositorio y enviarlo a las personas que queramos.

2) Realizar la siguiente actividad:
• Crear un repositorio.
- ☐ Dale un nombre al repositorio.
- ☐ Elige que 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

https://github.com/francobehm/trabajo-GitHub.git

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 estés 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.

https://github.com/francobehm/conflict-exercise.git