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 comunidad donde se puede crear y compartir repositorios de manera pública o privada
- **¿Cómo crear un repositorio en GitHub?**
  Desde la página GitHub
- **¿Cómo crear una rama en Git?**
  Con el comando git branch nombre-de-la-rama
- **¿Cómo cambiar a una rama en Git?**
  Con el comando git checkout nombre-de-la-rama
- **¿Cómo fusionar ramas en Git?**
  Se utiliza el comando git marge, primero nos movemos a la rama donde queremos que terminen los cambios, y luego hacemos merge de la rama en la que hicimos los cambios
- **¿Cómo crear un commit en Git?**
  Se crea con el comando git commit.
- **¿Cómo enviar un commit a GitHub?**
  git push nombre-de-la-rama
- **¿Qué es un repositorio remoto?**
  Es un lugar en la nube donde subís tu código para guardarlo

- **¿Cómo agregar un repositorio remoto a Git?**
  Se agrega con git remote add origin URL, donde URL es el link del repositorio que hayamos creado en GitHub

- **¿Cómo empujar cambios a un repositorio remoto?**
  Guardamos los cambios realizados en un commit y luego lo subimos a GitHub con git push

- **¿Cómo tirar de cambios de un repositorio remoto?**
  Utilizamos el comando git pull

- **¿Qué es un fork de repositorio?**
  Un fork es una copia de un repositorio que está en otra cuenta de GitHub.

- **¿Cómo crear un fork de un repositorio?**
  Debemos ingresar al repositorio GitHub del que queremos crear un fork y buscar la opción que nos permite hacerlo, nos va a preguntar si queremos forkearlo a nuestra cuenta y una vez aceptado vamos a tener una copia de ese proyecto en nuestra cuenta.

- **¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?**
  Hacemos un fork del proyecto, lo clonamos en nuestra compu, hacemos los cambios correspondientes y lo subimos con git push, vamos a nuestro repositorio de GitHub y buscamos la opción de "pull request", luego buscamos "New pull request", por último, seleccionamos "Create pull request" escribimos un mensaje con las modificaciones realizadas.

- **¿Cómo aceptar una solicitud de extracción?**
  Ingresamos a pull request, tendremos una lista de solicitudes pendientes y debemos ingresar a la que queramos revisar, verificamos los cambios realizados y si todo está bien hacemos click en "Marge pull request" y confirmamos el marge.

- **¿Qué es una etiqueta en Git?**
  Es un marcador que sirve para señalar un punto de interés en nuestro proyecto.

- **¿Cómo crear una etiqueta en Git?**
  Con el comando git tag nombre-de-la-etiqueta

- **¿Cómo enviar una etiqueta a GitHub?**
  Usamos git push nombre-de-la-etiqueta

- **¿Qué es un historial de Git?**
  Es la lista de todos los commits que se hicieron en un proyecto desde que se empezó a usar un Git

- **¿Cómo ver el historial de Git?**
  Utilizando el comando git log

- **¿Cómo buscar en el historial de Git?**
  Con el comando git log podemos buscar cambios específicos, por palabra en mensaje, por archivo especifico, por texto especifico

- **¿Cómo borrar el historial de Git?**
  Se utiliza el comando git rm nombre-archivo-o-carpeta

- **¿Qué es un repositorio privado en GitHub?**
  Es un proyecto en GitHub que solo vos y las personas que vos invites pueden ver o modificar.

- **¿Cómo crear un repositorio privado en GitHub?**
  Se debe iniciar sesión en GitHub, crear un nuevo repositorio y marcar la opción "Privado" antes de confirmarlo. De esta manera, solo el creador y las personas que él autorice podrán ver el contenido.

- **¿Cómo invitar a alguien a un repositorio privado en GitHub?**
  Se debe acceder al repositorio privado, ingresar a la sección de "Settings" y luego a "Collaborators" o "Manage access". Desde allí, se puede invitar a otros usuarios ingresando su nombre de usuario de GitHub y enviando la invitación correspondiente.

- **¿Qué es un repositorio público en GitHub?**
  Un repositorio público en GitHub es un proyecto cuyo contenido puede ser visto por cualquier persona.

- **¿Cómo crear un repositorio público en GitHub?**
  Se debe iniciar sesión en GitHub, crear un nuevo repositorio y seleccionar la opción "Público" antes de confirmarlo

- **¿Cómo compartir un repositorio público en GitHub?**
  Para compartir un repositorio público en GitHub, simplemente se debe copiar la URL del proyecto desde la barra de direcciones del navegador y enviarla a quien se desee. Al ser público, cualquier persona podrá acceder al contenido sin restricciones.

2) Realizar la siguiente actividad:

- 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:

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.