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 basada en la nube que permite a los desarrolladores almacenar, gestionar y compartir código utilizando el sistema de control de versiones Git.

- ¿Cómo crear un repositorio en GitHub?

Se puede crear desde la página de GitHub, o desde la consola de git.

- ¿Cómo crear una rama en Git?

Para crear una rama en git, se debe introducir el comando git branch con el nombre de la rama, entonces quedaría git branch ramita, por ejemplo.

- ¿Cómo cambiar a una rama en Git?

Para cambiar a una rama en git, se debe colocar el comando git checkout, si seguimos el ejemplo de la consigna anterior quedaría, git checkout ramita.

- ¿Cómo fusionar ramas en Git?

Para fusionar ramas en git se usa el comando git merge, y uno se debe parar en la rama donde quiere fusionar los cambios. Por ejemplo, me paro en la rama principal que se llama master, y quiero incluir los cambios que trabaje en ramita, entonces coloco git merge ramita.

- ¿Cómo crear un commit en Git?

Para crear un commit primero hay que cargar las modificaciones que uno ha realizado, esto se hace con git add ., luego se procede a aplicar el comando git commit con un mensaje, que se

realiza sumando -m al comando y luego el mensaje entre comillas, completo quedaría: git commit -m “Realizando un commit”.

- ¿Cómo enviar un commit a GitHub?

Luego de hacer el commit, se puede enviarlo a GitHub mediante el comando git push, el comando completo quedaría, git push origin master.

- ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión de un proyecto git que está alojada en un servidor externo, ya sea en Internet o en una red local.

- ¿Cómo agregar un repositorio remoto a Git?

Se utiliza el comando git remote add + la url del repositorio remoto.

- ¿Cómo empujar cambios a un repositorio remoto?

Esto se realiza con el comando git push origin master.

- ¿Cómo tirar de cambios de un repositorio remoto?

Esto se realiza con el comando git pull origin master, se puede colocar el nombre de cualquier rama.

- ¿Qué es un fork de repositorio?

Un fork de repositorio es una copia independiente de un repositorio original en GitHub o cualquier otra plataforma de control de versiones.

- ¿Cómo crear un fork de un repositorio?

Se accede al repositorio original en GitHub y se hace clic en fork, se selecciona el destino y listo.

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

Se debe trabajar desde un fork del repositorio original, preferentemente crear una rama nueva, realizar los cambios que se desee, luego entrar a la página del repositorio remoto en GitHub. Después selecciona el tu repositorio forkeado (donde están las modificaciones), finalmente hacer clic en compare and pull request.

- ¿Cómo aceptar una solicitud de extracción?

Para aceptar la solicitud, hay que dirigirse a pull Requests, luego, ir a la pestaña de conversation, hacer clic en review changes, después seleccionar approve y por último clic en Submit review.

- ¿Qué es una etiqueta en Git?

Una etiqueta en Git es un marcador que se utiliza para identificar un punto específico en el historial de commits de un repositorio.

- ¿Cómo crear una etiqueta en Git?

Esto se realiza con el comando git tag “Nombre etiqueta”.

- ¿Cómo enviar una etiqueta a GitHub?

Con el comando git push origin <nombre etiqueta>.

- ¿Qué es un historial de Git?

Un historial de Git es un registro completo y cronológico de todos los cambios realizados en un repositorio, representado por una secuencia de commits.

- ¿Cómo ver el historial de Git?

Con el comando git log.

- ¿Cómo buscar en el historial de Git?

Se puede buscar en el historial con distintos filtros que se aplican con distintos comandos, por ejemplo, por cantidad de commits “git log -n 3”, mostrará los últimos 3 commits, se puede filtrar por autor “git log –author=”Nombre del autor”” y así con otros filtros.

- ¿Cómo borrar el historial de Git?

Hay distintas formas de borrar el historial en Git, una es mover los commits que uno quiera a una nueva rama, luego borrar la rama main o master, por último renombrar la rama con el nombre main o master.

- ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un espacio de almacenamiento para proyectos al que solo pueden acceder usuarios específicamente autorizados.

- ¿Cómo crear un repositorio privado en GitHub?

Al crear un repositorio seleccionar la opción privado en la sección Visibility.

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

Ir al repositorio privado, clickear en settings, luego en Access y después en Collaborators, por último en Add people, ahí se puede seleccionar a quien se desee.

- ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un espacio de almacenamiento en la nube donde los desarrolladores pueden compartir su código de manera abierta y accesible para cualquier persona en Internet.

- ¿Cómo crear un repositorio público en GitHub?

Al crear un repositorio nuevo seleccionar público en visibilidad.

- ¿Cómo compartir un repositorio público en GitHub?

Para compartir un repositorio público basta con compartir el link del repositorio.

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.