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 en línea que permite alojar y gestionar repositorios de código utilizando Git, facilitando la colaboración entre desarrolladores.

- ¿Cómo crear un repositorio en GitHub?

  Para crear un repositorio en GitHub: accede a tu cuenta, haz clic en "New", asigna un nombre, configura opciones (público/privado, README) y presiona "Create repository".

- ¿Cómo crear una rama en Git?

  Usa el comando git branch nombre_de_rama para crear una nueva rama en Git

- ¿Cómo cambiar a una rama en Git?

Para cambiar a una rama en Git, usa el comando git checkout nombre_de_rama o git switch nombre_de_rama (en versiones más recientes de Git).

- ¿Cómo fusionar ramas en Git?

  ara fusionar ramas en Git, utiliza el comando git merge

- ¿Cómo crear un commit en Git?

  Usar git add para preparar los cambios.

  Ejecuta git commit -m "mensaje" para registrar el commit.

- ¿Cómo enviar un commit a GitHub?

  Asegúrate de estar en la rama correcta: git checkout <rama>.

  Usa git push origin <rama> para enviar el commit al repositorio remoto.

- ¿Qué es un repositorio remoto?

  Un repositorio remoto en Git es una copia de tu proyecto que se encuentra en un servidor en la nube o en otro equipo. Se utiliza para colaborar y compartir código con otros compañeros. Ejemplo: los repositorios en GitHub.

- ¿Cómo agregar un repositorio remoto a Git?

  Se usa este comando:

  git remote add origin <URL-del-repositorio>

- ¿Cómo empujar cambios a un repositorio remoto?

  Se usa este comando:

  git push origin <rama>

- ¿Cómo tirar de cambios de un repositorio remoto?

  Se usa este comando:

  git pull origin <rama>

- ¿Qué es un fork de repositorio?

Un fork es una copia de un repositorio que se hace en tu propia cuenta, generalmente desde plataformas como GitHub. Te permite experimentar o desarrollar sobre el código original sin afectar el proyecto principal. Es útil para colaborar o personalizar proyectos de otros desarrolladores.

- ¿Cómo crear un fork de un repositorio?

  Para crear un fork de un repositorio en GitHub:

  1. Ve al repositorio original que deseas copiar.
  2. Haz clic en el botón **Fork**, que aparece en la parte superior derecha.

  Esto hará una copia del repositorio en tu cuenta, listo para ser modificado.

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

  Para enviar una solicitud de extracción (pull request) en GitHub:

  1. Haz cambios en tu fork: Asegúrate de haber realizado los cambios necesarios en una rama.
  2. Ve al repositorio original: Accede al repositorio donde deseas enviar los cambios.
  3. Crea el pull request: Haz clic en Pull Requests y selecciona New Pull Request. Luego, elige la rama de tu fork con los cambios y sigue las instrucciones.

- ¿Cómo aceptar una solicitud de extracción?

  Para aceptar una solicitud de extracción (pull request) en GitHub:

  1. Ve a la sección **Pull Requests** del repositorio.
  2. Selecciona la solicitud que deseas aceptar.
  3. Haz clic en **Merge Pull Request** y confirma la acción.

- ¿Qué es una etiqueta en Git?

  Una etiqueta (o tag) en Git es un marcador que señala un punto específico en el historial del repositorio, como una versión de lanzamiento.

- ¿Cómo crear una etiqueta en Git?

  Usar este comando:

  git tag <nombre-de-la-etiqueta>

  Opcionalmente, agrega un mensaje:

  git tag -a <nombre-de-la-etiqueta> -m "Descripción de la etiqueta"

- ¿Cómo enviar una etiqueta a GitHub?

  Usar estos comandos:

  1. Empuja la etiqueta específica:

     git push origin <nombre-de-la-etiqueta>

  2. O empuja todas las etiquetas a la vez:

     git push origin --tags

- ¿Qué es un historial de Git?

  El historial de Git es el registro completo de todos los commits realizados en un repositorio. Muestra detalles como el autor, fecha, mensaje del commit y los cambios realizados. Puedes verlo con el comando:

  git log

- ¿Cómo ver el historial de Git?

  Usar el comando:

  git log

- ¿Cómo buscar en el historial de Git?

  Usa el comando gitpara buscar en el historial de Git

- ¿Cómo borrar el historial de Git?

  No se puede borrar el historial, pero puedo reiniciar el repositorio.

- ¿Qué es un repositorio privado en GitHub?

  Es un repositorio accesible solo por usuarios autorizados.

- ¿Cómo crear un repositorio privado en GitHub?

En la opción "Private" al crear el repositorio.

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

  Ir a "Settings" > "Manage access".

- ¿Qué es un repositorio público en GitHub?

  Es un repositorio accesible por cualquier persona.

- ¿Cómo crear un repositorio público en GitHub?

  Marca la opción "Public" al crear el repositorio.

- ¿Cómo compartir un repositorio público en GitHub?

  Comparte la URL del repositorio.

2) Realizar la siguiente actividad:

- Crear un repositorio.
  - Dale un nombre al repositorio. o 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

o Crear una Branch o Realizar

cambios o agregar un archivo o

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.

tp-2-programacion-git
https://github.com/Micaelapicco17/tp-2-programacion-git.git
conflict-exercise
https://github.com/Micaelapicco17/conflict-exercise.git