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 dónde podemos alojar nuestros repositorios. Es una comunidad en la que podemos compartirlos de forma pública o privada.

- ¿Cómo crear una rama en Git?

  Para trabajar en una rama paralela, podemos crear una nueva rama ejecutando el comando: git branch <nombre_que_queramos_ponerle>

- ¿Cómo cambiar a una rama en Git?

  Si queremos movernos a otra rama podemos ejecutar el comando:

  git checkout <nombre_de_la_rama>

- ¿Cómo fusionar ramas en Git?

Primero nos posicionamos en la rama que queremos que queden guardados los cambios y luego ejecutamos el siguiente comando: git merge <nombre_de_la_rama_que_tiene_el_contenidol>

- ¿Cómo crear un commit en Git?

  Para crear un commit ejecutamos el siguiente comando:

  git commit -m “<Descripción de lo realizado>”

- ¿Cómo enviar un commit a GitHub?

  Para enviar un commit ejecutamos el siguiente comando:

  Si es la primera vez:

  git push -u origin <nombre_de_la_rama>

  luego de la primera vez:

  git push

- ¿Qué es un repositorio remoto?

  Un repositorio remoto es un espacio en línea donde se guardan archivos o proyectos, facilitando el acceso y la colaboración.

- ¿Cómo agregar un repositorio remoto a Git?

  Para agregar un repositorio remoto ejecutamos el siguiente comando:

  git remote add origin <url>

- ¿Cómo empujar cambios a un repositorio remoto?

  Para empujar cambios a un repositorio remoto ejecutamos el siguiente comando:

  git push -u origin master (Si es la primera vez)

  git push (Luego de la primera vez)

- ¿Cómo tirar de cambios de un repositorio remoto?

  Para tirar cambios de un repositorio remoto ejecutamos el siguiente comando:

  git pull origin master

  git pull (si usamos -u en el push)

- ¿Qué es un fork de repositorio?

En Git, un “fork” es una copia de un repositorio creada en una cuenta diferente, lo que permite desarrollar cambios sin afectar al repositorio original.

- ¿Cómo crear un fork de un repositorio?

  Ingresamos a la plataforma Github con nuestro usuario y contraseña, buscamos el repositorio que queremos copiar y hacemos click en el botón “fork”.

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

  Para enviar una solicitud de extracción debemos ingresar al repositorio en GitHub, y en la pestaña “Pull requests”, hacer click en “New pull request”

  Luego seleccionamos la rama de base y la rama con los cambios y hacemos click en “Create pull request”

- ¿Cómo aceptar una solicitud de extracción?

  Para aceptar una solicitud de extracción debemos ingresar al repositorio en GitHub, y en la pestaña “Pull requests”, seleccionar el pull request pendiente. Luego hacemos click en “Merge pull request” y posteriormente confirmamos la fusión haciendo click en “Confirm merge”.

- ¿Qué es un etiqueta en Git?

  Las etiquetas sirven para poder identificar de manera única un commit específico, generalmente para marcar versiones importantes del proyecto.

- ¿Cómo crear una etiqueta en Git?

  Para crear una etiqueta en Git ejecutamos los siguientes comandos:

  Para crear un tag con mensaje:

  git tag -a <nombre> -m <mensaje>

  Para crear un tag en un commit específico:

  git tag -a <nombre hash> -m <mensaje>

- ¿Cómo enviar una etiqueta a GitHub?

  Para enviar una etiqueta en GitHub ejecutamos el siguiente comando:

  git push --tags

- ¿Qué es un historial de Git?

  El historial en Git se centra en los commits, que registran los cambios realizados en el proyecto. También incluye información sobre ramas y etiquetas. En

conjunto, permite rastrear y gestionar la evolución del código a lo largo del tiempo.

- ¿Cómo ver el historial de Git?

  Para ver el historial de los commits en Git, ejecutamos el comando:

  git log (para salir utilizamos la letra “q”)

  Para mostrar el historial de las referencias (movimientos de ramas y HEAD) ejecutamos el comando:

  git reflog

- ¿Cómo buscar en el historial de Git?

  Una manera práctica de buscar en el historial de Git es utilizar el siguiente comando:

  git log --decorate --all --graph –oneline

  Este comando muestra el historial de commits de forma compacta y visual

- ¿Cómo borrar el historial de Git?

  Para borrar el historial de Git, debemos crear una nueva rama sin historial usando el comando git checkout --orphan <nueva-rama>. Luego, agregamos todos los archivos con git add . , hacemos un commit con git commit -m "<Nuevo comienzo>", y eliminamos la rama principal con git branch -D main. Después, renombramos la nueva rama a "main" y forzamos el push con git push -f origin main. Así, eliminamos todo el historial de commits y solo queda el estado actual del proyecto.

- ¿Qué es un repositorio privado en GitHub?

  Un repositorio privado en GitHub es un repositorio cuyo contenido solo pueden ver y acceder las personas invitadas para colaborar. A diferencia de los repositorios públicos, solo los usuarios o equipos seleccionados tienen acceso al código.

- ¿Cómo crear un repositorio privado en GitHub?

  Para crear un repositorio privado en GitHub, debemos iniciar sesión y hacer clic en el "+" en la parte superior derecha. Luego, seleccionamos "New repository", completamos el nombre y la descripción del repositorio, y debajo de la descripción elegimos la opción "Private". Finalmente, hacemos clic en "Create repository", y de esta forma el repositorio será privado, accesible solo para las personas que invitemos.

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

  Para invitar a alguien a un repositorio privado en GitHub, ingresamos al repositorio, luego vamos a "Settings" y seleccionamos "Manage access". Hacemos clic en "Invite a collaborator", escribimos el nombre de usuario y lo agregamos. La persona recibirá la invitación para acceder al repositorio.

- ¿Qué es un repositorio público en GitHub?

  Un repositorio público en GitHub es un repositorio cuyo contenido está accesible para cualquier persona. Cualquier usuario puede ver, clonar y contribuir al código, siendo utilizado generalmente para proyectos de código abierto o para compartir trabajos con la comunidad.

- ¿Cómo crear un repositorio público en GitHub?

  Para crear un repositorio público en GitHub, debemos iniciar sesión y hacer clic en el "+" en la parte superior derecha. Luego, seleccionamos "New repository", completamos el nombre y la descripción del repositorio, y debajo de la descripción elegimos la opción "Public". Finalmente, hacemos clic en "Create repository", y de esta forma el repositorio será público, accesible para cualquier persona.

- ¿Cómo compartir un repositorio público en GitHub?

  Para compartir un repositorio público en GitHub, debemos ingresar al repositorio y hacer clic en el botón "Code" en la parte superior derecha. Luego, copiamos la URL que aparece en "Clone with HTTPS". Con esa URL, podemos compartir el repositorio por cualquier medio, y cualquier persona podrá acceder a él.

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

  https://github.com/JohnTheMano/primer-repo-git.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 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.

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