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?
  Comunidad online que sirve para guardar, buscar, copiar y compartir repositorios con otrxs colaboradxs.
- ¿Cómo crear un repositorio en GitHub?

  Abrir una cuenta en GitHub, arriba a la derecha clickear el signo “+” y seleccionar “New repository”.
- ¿Cómo crear una rama en Git?

  Se crea automáticamente al iniciar un proyecto, en caso de querer agregar una nueva rama al proyecto se utiliza el comando git branch <nombre>.
- ¿Cómo cambiar a una rama en Git?

  Con el comando git checkout <nombre de la rama>.
- ¿Cómo fusionar ramas en Git?

  Posicionarse sobre una de las ramas que se desea fusionar y escribir el comando git merge <otra rama a fusionar>.
- ¿Cómo crear un commit en Git?

  Con el comando git commit -m “y escribir el mensaje que se desea guadar”.

- ¿Cómo enviar un commit a GitHub?

  Con git push es posible subir los commits al repositorio remoto en GitHub.

- ¿Qué es un repositorio remoto?

  Versión de un proyecto que se sube a la nube mediante un servidor (GitHub) o de forma remota en una red (como una empresa).

- ¿Cómo agregar un repositorio remoto a Git?

  A través de la terminal ir hasta la dirección del repositorio local y utilizar comando git remote add <nombre_repo> <url_repo>.

- ¿Cómo empujar cambios a un repositorio remoto?

  Con el comando git push <nombre del repositorio> <nombre de la rama> se pueden empujar los cambios del repo local al remoto.

- ¿Cómo tirar de cambios de un repositorio remoto?
  Con el comando git pull <nombre del repositorio> es posible descargar los cambios del repositorio remoto y fusionarlos con el repo local (merge).

- ¿Qué es un fork de repositorio?

  Es una copia independiente de un repositorio en GitHub al propio repositorio personal.

- ¿Cómo crear un fork de un repositorio?

  Desde la página de GitHub se busca el repositorio de interés que se quiera copiar y desde el botón de la esquina superior derecha que dice “Fork” se presiona y reenvía para otorgarle un nombre a dicho repositorio.

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

  Una vez realizado el Fork, clonado el repositorio, habiendo hecho los cambios pertinentes y habiéndolos subido está todo listo para hacer un pull request a través de nuestro repositorio en GitHub.

- ¿Cómo aceptar una solicitud de extracción?

  Ir al enlace del repositorio en cuestión y hacer click en el apartado de Pull Request, analizar la propuesta y en caso de ser aprobada se mergea.

- ¿Qué es una etiqueta en Git?

  Funciona como un marcador de página de un libro. Marca una referencia de un commit específico en una parte del historial del código.

- ¿Cómo crear una etiqueta en Git?

  Con el comando git tag <nombre_de_version>.

- ¿Cómo enviar una etiqueta a GitHub?

  Para enviar o compartir una etiqueta se utiliza el comando git push <nombre_del_repositorio> <nombre_de_la_etiqueta>.

- ¿Qué es un historial de Git?

Un historial de Git es un registro de cada cambio realizado en un proyecto determinado, abarca desde su creación hasta la última modificación.

- ¿Cómo ver el historial de Git?

  Para ver el historial en Git es necesario ingresar el comando git log.

- ¿Cómo buscar en el historial de Git?

  Utilizar git log –grep <termino_a_buscar>, es posible hacer búsquedas por palabra, autor, fecha,etc. Para es necesario modificar el término –grep.

- ¿Cómo borrar el historial de Git?

  Es posible mediante el rm -rf .git

- ¿Qué es un repositorio privado en GitHub?

  Posee contenido visible sólo para el propietario y posibles colaboradores.

- ¿Cómo crear un repositorio privado en GitHub?

  Es necesario abrir la cuenta en GitHub, ir al perfil e ir arriba a la derecha donde se encuentra el signo “+” y seleccionar “New repository”, donde se podrá elegir la opción “Private”. Ir hasta abajo y clickear sobre “Create Repository”.

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

  Ubicarse en el enlace del repositorio que se desea compartir, seleccionar “settings” y a la izquierda se verá un apartado que de acceso a colaborades, una vez ahí se baja y se clickea sobre “Add People”.

- ¿Qué es un repositorio público en GitHub?
  Es un repositorio con contenido abierto a cualquier persona que quiera acceder a él.

- ¿Cómo crear un repositorio público en GitHub?

  De la misma forma que se crea un repositorio privado pero antes de ir a “Create Repository” seleccionar la opción de “Public”.

- ¿Cómo compartir un repositorio público en GitHub?

  De la misma forma que en el privado. El repositorio público tiene la característica también de poder compartirlo mediante link.

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

HECHO

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".

HECHO

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
```

HECHO

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"

HECHO

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"

HECHO

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.

HECHO

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"

HECHO

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

HECHO

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.

HECHO