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 basada en la nube que permite a los desarrolladores colaborar en proyectos, es similar a una red social para devs.
- ¿Cómo crear un repositorio en GitHub? Para crear un repositorio debemos contar con una cuenta y loguearnos, luego crear una carpeta del proyecto y situarnos en ella, iniciamos la carpeta con el comando “git init” y la agregamos a nuestro repositorio con el comando “git add .”
- ¿Cómo crear una rama en Git? Para ello utilizamos el comando git branch
- ¿Cómo cambiar a una rama en Git? Utilizamos el comando git checkout nombreDeLaRama
- ¿Cómo fusionar ramas en Git? con el comando git merge
- ¿Cómo crear un commit en Git? con el comando git commit -m .Para agregar un mesaje al commit
- ¿Cómo enviar un commit a GitHub?Utilizamos el comando git push.
- ¿Qué es un repositorio remoto?Es un repositorio en la nube como podria ser github.
- ¿Cómo agregar un repositorio remoto a Git? Utilizamos el comando git init .
- ¿Cómo empujar cambios a un repositorio remoto?con el comando git push .

- ¿Cómo tirar de cambios de un repositorio remoto?con el comando git pull .
- ¿Qué es un fork de repositorio? Es una copia de un proyecto que se crea en tu cuenta y esta cuenta es independiente del repositorio original.
- ¿Cómo crear un fork de un repositorio? No situamos sobre el repositorio que deseamos copiar y le damos clic al boton Fork que se encuentra en la parte superior derecha de la pagina del repositorio.

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?Nos situamos en el repositorio original y le damos clic en el boton pull request .
- ¿Cómo aceptar una solicitud de extracción? Damos clic en el boton Merge pull request.
- ¿Qué es un etiqueta en Git?Una etiqueta (o *tag*) en Git es una referencia que señala un punto específico en la historia de tu proyecto. Generalmente, se utiliza para marcar versiones importantes del código, como lanzamientos de software, versiones estables o hitos del desarrollo.
- ¿Cómo crear una etiqueta en Git? Utilizamos el comando git tag .
- ¿Cómo enviar una etiqueta a GitHub? Utilizamos el comando git push origin nombreDeLaEtiqueta .
- ¿Qué es un historial de Git? Es el registro de todos los cambios realizados en un proyecto, ordenados cronologicamente.
- ¿Cómo ver el historial de Git? Con el comando git log .
- ¿Cómo buscar en el historial de Git? Con el comando git log –grep=”palabra-clave”
    - ¿Cómo borrar el historial de Git? Utilizazmos el comando rm -rf .git .
- ¿Qué es un repositorio privado en GitHub? Es un repositorio configurado para que solo puedan acceder a el los usuarios autorizados.
- ¿Cómo crear un repositorio privado en GitHub? Al crear un repositorio seleccionamos la opcion de Privado.
- ¿Cómo invitar a alguien a un repositorio privado en GitHub? Nos situamos en el repositorio privado, damos clic en la pestaña de configuracion y en el menu de la izquierda selleccionamos Collaborators and teams y damos clic en el boton add people y aqui agregamos el email o nombre de usuario al cual le podemos dar un nivel de acceso.
- ¿Qué es un repositorio público en GitHub? Es un repositorio al que cualquier persona puede acceder, ver y clonar sin restricciones. En otras palabras es un proyecto abierto.
- ¿Cómo crear un repositorio público en GitHub? Creamo un nuevo repositorio y seleccionamos la opcion Public.
- ¿Cómo compartir un repositorio público en GitHub?Copiamos y compartimos la url.

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.

resolucion ejercicio 2 y 3 https://github.com/nicolasmdz/conflict-exercise.git