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 comunidad donde podemos compartir nuestros repositorios de forma publica o privada. Se dispone de un perfil donde podemos cargar nuestros repositorios, clonar repos, interactuar con otros usuarios, entre otros. Nos permite controlar las versiones de nuestros proyectos de forma remota.
- ¿Cómo crear un repositorio en GitHub? Ingresamos en https://github.com/, nos logueamos, hacemos click en el botón "+" en la parte superior derecha y seleccionamos la opción de "New Repository". configuramos a nuestro gusto con nombre, descripción etc. y luego hacemos click en "Create Repository", un botón verde en la parte de abajo derecha
- ¿Cómo crear una rama en Git? Para crear una rama nueva usamos el comando git branch nombre-rama (cambiamos el "nombre-rama" por el nombre que queremos darle a esta rama)
- ¿Cómo cambiar a una rama en Git? Utilizamos el comando git checkout rama (cambiamos el "rama" por el nombre de la rama a la cual queremos cambiarnos)
- ¿Cómo fusionar ramas en Git? Esto se realiza con el comando git merge rama (cambiamos el "rama" por el nombre de la rama que queremos fusionar a la rama actual donde estamos parados)
- ¿Cómo crear un commit en Git? Utilizamos git add . para agregar los cambios a la rama y luego usamos el git commit -m "Mensaje descriptivo"
- ¿Cómo enviar un commit a GitHub? Utilizamos el comando git push origin nombre-rama (por lo general el primer commit se envia a main o master"
- ¿Qué es un repositorio remoto? Es una versión de tu repositorio de git que esta alojado en un servidor o en la nueve, por ejemplo GitHub
- ¿Cómo agregar un repositorio remoto a Git? Esto se realiza con el comando git remote add origin link-repo si queremos agregar un repo local y vincularlo a un repo remoto o con el git clone link-repo si queremos clonar un repo remoto a nuestro entorno local
- ¿Cómo empujar cambios a un repositorio remoto? Esto se realiza con el comando git push origin nombre-rama
- ¿Cómo tirar de cambios de un repositorio remoto? Esto se realiza con el comando git pull origin nombre-rama
- ¿Qué es un fork de repositorio? Es una copia de un repositorio que se encuentra en tu cuenta de GitHub y esta separa del repositorio original.
- ¿Cómo crear un fork de un repositorio? Acceder al repositorio del cual se quiere hacer el fork, hacer click en el botón fork arriba a la derecha, seleccionar la cuenta para alojar el fork, esperar que se cree y ya estará disponible para trabajar con él.

Luego de hacer los cambios con el git add, git commit y git push en las ramas deseadas, ingresar al repositorio y se habrá habilitado un botón que dice "Compare & pull request", le hacemos click, rellenamos los detalles del PR, revisamos y enviamos la solicitud con el botón "Create pull request"

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
- ¿Cómo aceptar una solicitud de extracción? En el repo, en la pestana "Pull request" veremos la solicitud, ingresamos en ella, revisamos la misma, si todo esta bien la aprobamos con el botón "Approve" y luego se podrá proceder a hacer click en el botón Merge pull request", procedemos con el botón "Confirm merge" y por ultimo borramos la rama que ya no usaremos con el botón "Delete branch"
- ¿Qué es un etiqueta en Git? Una tag es un marcador utilizado para identificar un punto especifico en el historial de commits.
- ¿Cómo crear una etiqueta en Git? Seleccionamos la rama y estado de correcto en el historial de git donde deseemos agregar la tag y usamos el comando git tan nombre-tag
- ¿Cómo enviar una etiqueta a GitHub? Usamos el comango git push --tags
- ¿Qué es un historial de Git? Es un registro de todos los cambios realizados en el repositorio a lo largo del tiempo
- ¿Cómo ver el historial de Git? Se utiliza el comando git log
- ¿Cómo buscar en el historial de Git? Se usa el comando git log seguido de "--criterio-de-busqueda" donde podria ser --grep="Mnesaje del commit" o --author="Nombre del autor del commit".
  - ¿Cómo borrar el historial de Git? Realmente no se pude borrar completamente ya que esta diseñado para ser un registro permanente pero se puede reescribir el historial usando el comando git rebase -i HEAD~n donde n es el numero de commit que se deseen reescribir
- ¿Qué es un repositorio privado en GitHub? Es un repo restringido para que solo las personas que el dueño del repo elijan puedan verlo
- ¿Cómo crear un repositorio privado en GitHub? Es igual a crear un desde el inicio pero al momento de configurarlo previo a hacer click en "Create repository" se selecciona la opcion de private antes de crearlo.
- ¿Cómo invitar a alguien a un repositorio privado en GitHub? Una vez creado el repo se va la pestaña "Settings", luego en la pestaña "Collaborators and teams" se selecciona a los usuarios que pueden tener acceso. y se les envía la invitación con el "Send Invitation"
- ¿Qué es un repositorio público en GitHub? Es un repo al cual cualquier persona tiene acceso de explorar y realizar colaboraciones
- ¿Cómo crear un repositorio público en GitHub? Es igual a crear un desde el inicio pero al momento de configurarlo previo a hacer click en "Create repository" se selecciona la opcion de public antes de crearlo.
- ¿Cómo compartir un repositorio público en GitHub? Se compartir con el link del repo simplemente o desde el boton "Code" donde te da más opciones para compartirlo

2) Realizar la siguiente actividad: LINK DEL REPO: https://github.com/Dario-Cabrera/TP2.git

- 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". LINK DEL REPO: https://github.com/Dario-Cabrera/-conflict-exercise-Cabrera-Dario.git

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.