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? Plataforma que permite alojar repositorios en internet para poder trabajar con otras personas en conjunto y sirve de backup para tu repositorio local.
- ¿Cómo crear un repositorio en GitHub? Iniciar sesion, apretar el botón new y ajustar las configuraciones que te ofrece si es uno nuevo, si queremos subir un local debemos inicializar git en nuestra computadora y realizar un push habiendo elegido como remoto nuestro github.
- ¿Cómo crear una rama en Git? Por comando se usa: git checkout -b nuevaRama, en la plataforma se puede crear pero luego hay que hacer git fetch para poder traerla al local.
- ¿Cómo cambiar a una rama en Git? git checkout otraRama
- ¿Cómo fusionar ramas en Git? git checkout master, git pull origin master (para asegurarte de que esté al dia) git merge otraRama (previamente actualizada) y git push para actualizar el remoto.
- ¿Cómo crear un commit en Git? git add . (Agrega todos los cambios) - git commit -m 'Realizo un commit.'
- ¿Cómo enviar un commit a GitHub? git push origin ramaEnUso
- ¿Qué es un repositorio remoto? Es un repositorio que está alojado en internet, por lo cual, se puede acceder desde otras computadoras y trabajar en equipo.
- ¿Cómo agregar un repositorio remoto a Git? git remote add origin (origin es un nombre de uso comun pero puede ser diferente) (url)
- ¿Cómo empujar cambios a un repositorio remoto? git push origin ramaEnUso (si es una rama propia, no tiene por qué tener cambios en el remoto que no están en mi local, pero previamente hacer siempre un pull de master.)
- ¿Cómo tirar de cambios de un repositorio remoto? git pull origin master (master o la rama que te interesa)
- ¿Qué es un fork de repositorio? Es una copia de un repositorio que nos interese compartir en nuestro perfil o modificar sin alterar el original.
- ¿Cómo crear un fork de un repositorio? Desde un botón en la página web.

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
  La rama tiene que estar actualizada con master, botón pull request, ajustar commits y realizar la petición. Si se acepta, merge.
- ¿Cómo aceptar una solicitud de extracción?
  Hay un botón para aceptar la solicitud. El merge puede ser realizado por el aprobador mismo pero depende de la configuración, también puede hacerlo quien lo solicita.
- ¿Qué es un etiqueta en Git?
  Es un objeto que permite marcar un commit en específico.
- ¿Cómo crear una etiqueta en Git? Con el comando: git tag (nombre)
- ¿Cómo enviar una etiqueta a GitHub? git push origin (nombre)
- ¿Qué es un historial de Git? Muestra las versiones anteriores de un archivo.
- ¿Cómo ver el historial de Git? Click derecho, git history en VSC.
  git log en comandos, se puede elegir una cantidad agregando guion y numero.
- ¿Cómo buscar en el historial de Git? git log puede acompañarse de: --grep="Case" si queremos encontrar un commit relacionado con la palabra Case. - --CaseTriggerHandler.cls para un archivo en especifico y --author="DGHISIO" para filtrar por programador. Rangos de fechas usando --since y --until
- ¿Cómo borrar el historial de Git?
  git reset
- ¿Qué es un repositorio privado en GitHub?
  Es un repositorio que no está disponible para cualquier usuario de github
- ¿Cómo crear un repositorio privado en GitHub?
  Al momento de crear un repositorio te muestra las opciones y debes seleccionar si queres que sea publico o privado
- ¿Cómo invitar a alguien a un repositorio privado en GitHub?
  Se agrega como colaborador, la persona tiene que tener una cuenta de GitHub, lo agregas con el nombre de usuario.
- ¿Qué es un repositorio público en GitHub?
  Está disponible para cualquier usuario que desee acceder.
- ¿Cómo crear un repositorio público en GitHub?
  De la misma manera que el otro.
- ¿Cómo compartir un repositorio público en GitHub?
  Compartiendo la URL del repositorio público.

2) Realizar la siguiente actividad:

https://github.com/daianaghisio/Nuevo_Repositorio.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:

https://github.com/daianaghisio/Otro_Repositorio.git

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.