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?
  Es una plataforma de desarrollo colaborativo basada en Git que permite alojar, gestionar y compartir código. Facilita el trabajo en equipo con herramientas como repositorios remotos, control de versiones, issues y pull requests.
- ¿Cómo crear un repositorio en GitHub?
  - Iniciar sesión en GitHub.
  - Clickear en el botón New o ir a Repositories → New.
  - Dar un nombre para el repositorio.
  - Elige si será público o privado.
  - Opcional: Agrega un README.md..
  - Hacer clic en Create repository.
- ¿Cómo crear una rama en Git?

  Git branch nombre-del-branch
- ¿Cómo cambiar a una rama en Git?

  Git checkout nombre-del-branch
- ¿Cómo fusionar ramas en Git?

  Estando en la main insertar el siguiente comando:

  Git merge nombre-del-branch

- ¿Cómo crear un commit en Git?

  Git add .

  Git commit –m “Descripción del cambio”

- ¿Cómo enviar un commit a GitHub?

  Primera vez:

  git push -u origin master

  Despues usar:

  git push

- ¿Qué es un repositorio remoto?

  Es una versión alojada en un servidor, como GitHub, que permite colaborar con otros.

- ¿Cómo agregar un repositorio remoto a Git?

  git remote add origin url

- ¿Cómo empujar cambios a un repositorio remoto?

  git push -u origin master

- ¿Cómo tirar de cambios de un repositorio remoto?

  git pull origin master

  git pull(Si usamos -u en el push)

- ¿Qué es un fork de repositorio?

  Un fork es una copia de un repositorio de otra persona en tu cuenta de GitHub. Permite modificarlo sin afectar el original.

- ¿Cómo crear un fork de un repositorio?

  1. Ve al repositorio en GitHub.
  2. Haz clic en el botón Fork (arriba a la derecha).
  3. Ahora tendrás una copia en tu cuenta.

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

  1. Sube tus cambios con git push origin url
  2. Ve a GitHub, entra a tu repositorio.
  3. Haz clic en Compare & pull request.
  4. Agrega una descripción y envía el pull request.

- ¿Cómo aceptar una solicitud de extracción?

  1. Entra al repositorio en GitHub.
  2. Ve a la pestaña Pull Requests.
  3. Abre la solicitud y revisa los cambios.
  4. Haz clic en Merge pull request.

- ¿Qué es un etiqueta en Git?

  Una etiqueta (tag) es una referencia a un commit específico, útil para marcar versiones.

- ¿Cómo crear una etiqueta en Git?

  Crear tag: git tag nombre

  Crear tag con mensaje: git tag -a nombre -m mensaje

  Crear tag en commit específico: git tag -a nombre hash -m mensaje

- ¿Cómo enviar una etiqueta a GitHub?

  git push origin nombre (ese tag especificamente)
  git push –tags (todos los tags)

- ¿Qué es un historial de Git?

  El historial de Git muestra todos los commits realizados en el repositorio.

- ¿Cómo ver el historial de Git?

  git log Ver commits (qpara salir)

  git log --oneline Ver commits(una línea c/u)

  git log --decorate --all --graph --oneline Ver commits (graficado)

  git log ramaB..ramaA Commits en ramaA que no estén en ramaB

  git log --follow archivo Commits donde el archivo cambió

- ¿Cómo buscar en el historial de Git?

  git log --follow archivo

  - ¿Cómo borrar el historial de Git?

    rm -rf .git

    git init

- ¿Qué es un repositorio privado en GitHub?

  Un repositorio privado solo puede ser visto por el dueño y colaboradores invitados.

- ¿Cómo crear un repositorio privado en GitHub?
  - Iniciar sesión en GitHub.
  - Clickear en el botón New o ir a Repositories → New.
  - Dar un nombre para el repositorio.
  - Elige privado.
  - Opcional: Agrega un README.md..
  - Hacer clic en Create repository.

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

  1. Entra al repositorio.
  2. Ve a Settings > Collaborators.
  3. Agrega el nombre de usuario y envía la invitación.

- ¿Qué es un repositorio público en GitHub?

  Un repositorio público es accesible para cualquier persona en GitHub.

- ¿Cómo crear un repositorio público en GitHub?
  - Clickear en el botón New o ir a Repositories → New.
  - Dar un nombre para el repositorio.
  - Elige publico.
  - Opcional: Agrega un README.md..
  - Hacer clic en Create repository.

- ¿Cómo compartir un repositorio público en GitHub?

  Se puede compartir enviando el enlace del repositorio.

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/FedericoAlmuina/Tabajo-practico-2

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/FedericoAlmuina/conflict-exercise