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 de desarrollo colaborativo** para alojar proyectos utilizando el sistema de **control de versiones Git**. Se utiliza principalmente para la creación de código fuente de programas de ordenador.
  El principal servicio que ofrece GitHub son los **repositorios** en donde los usuarios pueden almacenar el código de sus aplicaciones, ya sea de forma pública o privada

- ¿Cómo crear un repositorio en GitHub?

  Un repositorio es como una carpeta que contiene elementos relacionados, como archivos, imágenes, vídeos o incluso otras carpetas. Normalmente, un repositorio agrupa los elementos que pertenecen al mismo "proyecto" o aquello en lo que trabajes.

  Crear una cuenta personal

  Seleccionar **+** --> se abre un cuadro de dialogo

  Seleccionar **new repository**

  Poner nombre del repositorio

  Descripción

Si es publico o privado

Agregar archivo readme

Click en **create repository**

- ¿Cómo crear una rama en Git?

  En la terminal poner: **git branch nombre-de-rama-creada**

- ¿Cómo cambiar a una rama en Git?

  git checkout nombre-de-rama-creada

- ¿Cómo fusionar ramas en Git?

  Parado en la rama que queremos que quede (ej: main)

  git merge nombre-de-rama-creada

- ¿Cómo crear un commit en Git?

  Git considera cada commit un punto de cambio o "punto de guardado". Es un punto del proyecto al que puedes volver si encuentras un error o se quiere hacer un cambio.

  En la terminal **git commit –m " texto que sirva de referencia a los cambios"**

- ¿Cómo enviar un commit a GitHub?

  En la terminal : git push origin

- ¿Qué es un repositorio remoto?

  Los repositorios remotos son versiones de tu proyecto que están hospedadas en Internet o en cualquier otra red.

- ¿Cómo agregar un repositorio remoto a Git?

  Para agregar un repositorio remoto nuevo, use el comando git remote add en el terminal, dentro del directorio donde está almacenado su repositorio.

  El comando git remote add toma dos argumentos:

  Un nombre remoto, por ejemplo, origin

  Una dirección URL remota, por ejemplo, https://github.com/NUESTRO/REPOSITORIO.git

En la terminal: **git remote add origin https://github.com/NUESTRO/REPOSITORIO.git**

- ¿Cómo empujar cambios a un repositorio remoto?

  Git push origin main

- ¿Cómo tirar de cambios de un repositorio remoto?

  git pull

- ¿Qué es un fork de repositorio?

  Fork (bifurcacion) es una copia de un repositorio. Esto es útil cuando se quiere contribuir al proyecto de otra persona o iniciar tu propio proyecto basado en el suyo.

- ¿Cómo crear un fork de un repositorio?

  Parado en el repositorio que queremos hacer fork, hacemos click en **fork** (arriba a la derecha)

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
  Una vez realizados los cambios y verificados se realiza el pedido para que revisen los cambios
  Se suben los cambios a nuestro repositorio en GitHub y hacemos click en el botón "Pull request".
  Detallar todo lo necesario de lo que se ha hecho y despues envíar el pull request.

- ¿Cómo aceptar una solicitud de extracción?

  En el nombre del repositorio, hacer click en pull requests.

  Hacer click en la que quiera revisar.

  Hacer click en Archivos cambiados.

  Revisar los cambios en la solicitud. clic en Revisar cambios.

  Seleccionar **Aprobar** para aprobar la combinación de los cambios propuestos

  Click en Enviar revisión.

- ¿Qué es un etiqueta en Git?

  Git tiene la posibilidad de etiquetar puntos específicos del historial como importantes.

  Una etiqueta es muy parecido a una rama que no se bifurca , es un puntero con nombre (etiqueta) a un commit específico

- ¿Cómo crear una etiqueta en Git?

  Etiqueta ligera: git tag v1

  Etiqueta anotada: git tag –a v1 –m “versión 1”

- ¿Cómo enviar una etiqueta a GitHub?

  El comando git push no transfiere las etiquetas a los servidores remotos. Se debe enviar las etiquetas de forma explícita al servidor luego de que se hayan creado. Este proceso es similar al de compartir ramas remotas.

  Ejecutar git push origin [etiqueta]

  O con git push origin master –tag -- > se envían todas las etiquetas

- ¿Qué es un historial de Git?

  Muestra la historia de los cambios y commits

- ¿Cómo ver el historial de Git?

  Con el comando **git log**

  También se puede usar: **git log –p** muestra el detalle de cada commit

- ¿Cómo buscar en el historial de Git?

  Con **git log –p -2** : muestra los últimos dos

  Hay muchos parámetros que filtran la búsqueda :

  Git log --author “sermass” por autor

  Git log -- since=”2022-07-03” por fecha

- ¿Cómo borrar el historial de Git?

  Para deshacer cambios que se hayan hecho:

  git commit –amend

  Si se hace commit y luego te das cuenta que olvidaste preparar los cambios de un archivo que querías incluir en esta confirmación, puedes hacer lo siguiente:

  $ git commit -m ‘commit inicial’

  $ git add archivo_olvidado

  $ git commit –amend

Queda una sola confirmación - la segunda confirmación reemplaza el resultado de la primera.

- ¿Qué es un repositorio privado en GitHub?

  Un repositorio que restringe quien tiene acceso a él. Pueden acceder el creador y las personas que el comparta acceso explícitamente

- ¿Cómo crear un repositorio privado en GitHub?

  Cuando se crea el repositorio , se hace click en la opción privado

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

  Dentro del repositorio, en la solapa setting ir a colaboradores, te pide que coloques tu clave de acceso y adicionar gente (people)

- ¿Qué es un repositorio público en GitHub?

  Es un repositorio accesible para todo el mundo en internet

- ¿Cómo crear un repositorio público en GitHub?

  Cuando se crea el repositorio , se hace click en la opción publico

- ¿Cómo compartir un repositorio público en GitHub?

Dentro del repositorio, en la solapa setting ir a colaboradores (no pide tu clave de acceso) y adicionar gente (people)

2) Realizar la siguiente actividad: https://github.com/sermass/repo_local

- 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/sermass/repo-conflict

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.