Práctico 2: Git y GitHub

NOGUEIRA DAMIÁN IGNACIO

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 online que permite almacenar, gestionar y controlar versiones de código. Dicho de otra manera, es un “controlador de versiones”.

- ¿Cómo crear un repositorio en GitHub?

  Para crear un repositorio en GitHub, debemos crear una cuenta e iniciar sesión en la plataforma; luego debemos dirigirnos a la esquina superior derecha y clickear el símbolo “+”, y seleccionar “New Repository”, elegir un nombre para nuestro repositorio, optar si será del tipo “público” o “privado”, y presionar en “créate repository”.

- ¿Cómo crear una rama en Git?

Para crear una rama o branch en Git, podemos acceder a la terminal de nuestro editor de código, escribir el comando “git branch” seguido del nombre elegido para la nueva rama. Ejemplo: git branch nueva_rama.

- ¿Cómo cambiar a una rama en Git?

  Para cambiar de una rama a otra, debemos ingresar en la terminal el comando “git checkout” seguido del nombre de la rama a la que queremos dirigirnos. Ejemplo: git checkout nombre_rama.

- ¿Cómo fusionar ramas en Git?

  Para fusionar ramas, debemos ingresar en la terminal el comando “git merge” seguido del nombre de la rama que queremos fusionar (aquí debemos tener presente en qué rama nos encontramos al momento de fusionar, ya que será la rama que se fusione con la que elegimos, generalmente fusionamos la rama main o master con una rama secundaria). Ejemplo: git merge nombre_rama.

- ¿Cómo crear un commit en Git?

  Para crear un commit primero debemos ingresar en la terminal el comando “git status” para verificar si hubo modificaciones y cuáles. Luego debemos ingresar el comando “git add .” para agregar todos los archivos modificados. Finalmente ingresar el comando “git commit” seguido de “-m” para agregar un mensaje (aunque esto es opcional).

  Ejemplo: git status

  git add

  git commit –m “agregamos un mensaje”.

- ¿Cómo enviar un commit a GitHub?

  Para enviar un commit a GitHub, primero debemos crear el commit como expliqué anteriormente. Luego en la terminal debemos ingresar el comando “git push origin” seguido del nombre de la rama que estamos trabajando. Ejemplo: git push origin nombre_rama (generalmente es main).

  Pero si es la primera vez que enviamos un commit debemos ingresar el comando “git push –u origin” seguido del nombre de la rama. Ejemplo: git push –u origin main.

- ¿Qué es un repositorio remoto?

  Un repositorio remoto es una versión del repositorio que se encuentra alojada en un servidor externo, (como GitHub) en vez de en una computadora local. Nos permiten que varios desarrolladores accedan de forma remota y trabajen en un

proyecto, colaborando en los cambios realizados, sin necesidad de estar físicamente en un mismo lugar.

- ¿Cómo agregar un repositorio remoto a Git?

  Para agregar un repositorio a Git, primero debemos ingresar en la terminal el directorio de nuestro proyecto, una vez posicionados ahí, debemos ingresar el comando “git remote add” seguido de un nombre para el repositorio remoto (generalmente se usa origin) y agregar la URL del repositorio remoto. Ejemplo: git remote add origin https://github.com/damiannogueira/repositorio.git.

- ¿Cómo empujar cambios a un repositorio remoto?

  Para empujar cambios a un repositorio remoto debemos ingresar en la terminal el comando “git push” seguido del nombre del repositorio (origin generalmente) seguido del nombre la rama en la que estamos. Ejemplo: git push origin nombre_rama

- ¿Cómo tirar de cambios de un repositorio remoto?

  Para tirar u obtener los últimos cambios de un repositorio remoto, debemos ingresar en la terminal el comando “git pull” seguido del nombre del repositorio (origin) y el nombre de la rama. Ejemplo (rama main): git pull origin main.

- ¿Qué es un fork de repositorio?

  Un fork de repositorio es una copia de un repositorio que se crea en tu propia cuenta (por ejemplo de GitHub), pero vinculada al repositorio original. Nos permite hacer cambios en el proyecto de forma independiente sin afectar el proyecto original, y luego proponer esos cambios al repositorio original mediante un pull request**.**

- ¿Cómo crear un fork de un repositorio?

  Para crear a un fork de un repositorio, debemos dirigirnos a dicho repositorio (por ejemplo en GitHub), en la parte superior derecha hacer click en el botón “fork” y elegir dónde queremos crearlo, si en nuestra cuenta personal o en la de una organización para la que estemos trabajando. Para finalizar debemos clonar el fork a nuestra computadora ingresando en la terminal el comando “git clone” seguido de la url del fork creado. Ejemplo: git clone https://github.com/utn/repositorio-fork.git

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

  Para enviar una solicitud de extracción debemos dirigirnos al repositorio, y clickear el botón “pull requests”, luego en “new pull request”, luego click en “create pull request”.

- ¿Cómo aceptar una solicitud de extracción?

  Para aceptar una solicitud de extracción (pull request), debemos ser un colaborador o tener permisos de administración en el repositorio donde se envió la solicitud. Dirigirnos al repositorio donde se ha enviado el pull request, clickear el botón “Pull requests” y allí seleccionar el pull request que queremos aceptar; luego de realizar una revisión debemos clickear en “Merge pull request” y luego “Confirm merge”.

- ¿Qué es una etiqueta en Git?

  Una etiqueta en Git es una referencia que se utiliza para marcar puntos específicos en el historial de commits. Se utilizan para señalar versiones importantes del proyecto.

- ¿Cómo crear una etiqueta en Git?

  Para crear una etiqueta en Git, debemos ingresar en la terminal el comando “git tag” seguida del nombre elegido para la etiqueta. Ejemplo: git tag v1.0.

- ¿Cómo enviar una etiqueta a GitHub?

  Para enviar una etiqueta a GitHub, luego de crearla en el repositorio local, debemos ingresar en la terminal el comando “git push” seguido del nombre del repositorio y de la etiqueta. Ejemplo: git push origin v1.0.

- ¿Qué es un historial de Git?

  El historial de Git es un registro completo de todos los cambios realizados en un repositorio, incluye información detallada sobre cada commit. Estos commit contienen un conjunto de cambios en los archivos, junto con un mensaje descriptivo, la fecha, el autor y un identificador único llamado hash.

- ¿Cómo ver el historial de Git?

  Para ver el historial de Git, debemos ingresar en la terminal el comando “git log”.

- ¿Cómo buscar en el historial de Git?

  Para buscar en el historial de Git tenemos diferentes comandos que podemos ingresar en la terminal:

  Buscar por mensaje de commit: git log --grep="texto a buscar"

  Buscar por archivo específico: git log nombre-del-archivo

  Buscar por autor: git log --author="nombre del autor"

  Buscar por fecha: git log --since="fecha de inicio" --until="fecha de fin"

Buscar cambios específicos (diff): git log -p

Buscar por hash de commit: git log <commit-hash>

Buscar patrones de texto en el historial de commits: git log --grep="expresión regular"

- ¿Cómo borrar el historial de Git?

  Borrar el historial de Git es una tarea delicada, ya que modifica el historial de commits del repositorio. Encontré varias maneras de llevarlo a cabo:

  Eliminar un commit específico: Usamos el comando “git revert <commit-hash>” para deshacer cambios sin eliminar el commit.

  Eliminar commits recientes: Usamos el comando “git reset --soft HEAD~n” o “git reset --hard HEAD~n” para eliminar commits.

  Reescribir el historial: Usamos el comando “git rebase -i HEAD~n” para modificar varios commits en el historial.

  Eliminar una rama y su historial: Usamos el comando “git branch -D nombre_rama” para eliminar la rama localmente.

  Reescribir el historial completo: Usamos el comando “git filter-branch” o “git filter-repo” para eliminar información de todo el repositorio.

- ¿Qué es un repositorio privado en GitHub?

  Un repositorio privado en GitHub es un tipo de repositorio que solo puede ser accedido por un grupo selecto de usuarios o colaboradores. Está restringido a los usuarios que han sido invitados explícitamente o que tienen permisos para acceder a él.

- ¿Cómo crear un repositorio privado en GitHub?

  Para crear un repositorio privado en GitHub lo hacemos de la misma manera que expliqué anteriormente, con la diferencia que debemos configurar la sección “Visibility”, como “Private”. Luego podemos agregar los permisos para cada colaborador que tendrá acceso.

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

  Para invitar a alguien a un repositorio privado en GitHub, debemos ser administradores del mismo. Debemos acceder al menú “Settings”, luego “Collaborators and Teams” y “Add people”. Luego podemos brindarle permiso a modo de solo lectura, escritura o administrador.

- ¿Qué es un repositorio público en GitHub?

  Un repositorio público en GitHub es un tipo de repositorio que es accesible para cualquier usuario de la red, que disponga de una cuenta de GitHub, y sin necesidad de permisos especiales.

- ¿Cómo crear un repositorio público en GitHub?

  Para crear un repositorio público en GitHub lo hacemos de la misma forma que mencioné anteriormente, solo que debemos configurar la visibilidad “Visibility” como “Public” para que cualquier usuario pueda tener acceso al mismo.

- ¿Cómo compartir un repositorio público en GitHub?

  Para compartir un repositorio público en GitHub, solo necesitamos disponer del enlace de nuestro repositorio. Ejemplo: https://github.com/damian/nombre-del-repositorio; o bien utilizando la opción “share”.

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

Link a mi repositorio de GitHub: https://github.com/damiannogueira/TP2-Ejercicio2.git

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.

Link a mi repositorio de GitHub: https://github.com/damiannogueira/conflict-exercise.git