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 alojamiento colaborativo, donde atravez del sistema de control de versiones Git, podemos almacenar compartir y colaborar en proyectos de software. Vale la pena destacar que Git y GitHub no son lo mismo.

• ¿Cómo crear un repositorio en GitHub?

Debemos tener una cuenta en GitHub, una vez dentro de nuestra cuenta debemos dirigirnos a nuestro perfil (profile). En la barra debajo de nuestro nombre encontraremos un dashboard donde dirá Repositories, allí debemos ingresar. Una vez nos mostrará nuestro repositorios ya creados, allí mismo encontraremos un botón verde el cual dice New. Completando con nombre y definiendo si queremos que sea publico o privado, ya creamos nuestro repositorio.

• ¿Cómo crear una rama en Git?

Desde la consola del editor de código utilizo el comando git branch nombreDeRama.

• ¿Cómo cambiar a una rama en Git?

Con el comando git checkout nombreDeRama le estoy indicando que me cambie a la rama NombreDeRama

• ¿Cómo fusionar ramas en Git?

Debo posicionarme en una rama, teniendo en cuenta que lo que está en la otra rama se volcará en la rama que estoy actualmente. Por ejemplo, me posiciono en la rama main. Ejecuto el comando git merge nombreDeRama, esto hace que me traíga lo nuevo de NombreDeRama a main.

• ¿Cómo crear un commit en Git?

En este caso primero lo que debemos hacer es agregar los cambios, por lo cual ejecutamos el comando git add . , luego para realizar el commit ejecutamos git commit -m “Mensaje descriptivo del commit”

• ¿Cómo enviar un commit a GitHub?

Para enviar el commit al repositorio remoto, se debe ejecutar el comando git push origin main (si nos encontramos en la rama main).

• ¿Qué es un repositorio remoto?

Es una copia o versión del proyecto que tenemos localmente, pero almacenado en la nube.

• ¿Cómo agregar un repositorio remoto a Git?

En este caso debemos traernos la url del repositario que creamos en GitHub, luego ejecutar en la consola, posicionados en nuestro repositorio local el comando git remote add URL_DEL_REPOSITORIODEGITHUB

• ¿Cómo empujar cambios a un repositorio remoto?

Una vez realizado el commit, empujamos el cambio con el comando git push -u origin main

• ¿Cómo tirar de cambios de un repositorio remoto?

En caso debemos pullear los cambios del repo remoto, esto lo hacemos ejecutando el comando git pull origin main.

• ¿Qué es un fork de repositorio?

Un fork es una copia de un repositorio de otra persona que hago en mi cuenta de GitHub donde voy a poder modificarlo sin afectar el original que no me pertenece.

• ¿Cómo crear un fork de un repositorio?

Me debo posicionar en el repositorio, allí en el parte superior me encuentro con el botón fork, al hacer click me deja poner una descripción, y luego de confirmarlo ya se encuentra como un repositorio propio en mi cuenta.

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

Debo dirigirme al apartado de Pull Request del repositorio, allí encontramos el botón New Pull Request, para lo que debemos confirmar finalmente con Create Pull Request.

• ¿Cómo aceptar una solicitud de extracción?

Dentro de nuestro profile, nos dirigimos al apartado de Pull Request, allí debo seleccionar la solicitud, y confirmar con el botón Merge Pull Request.

• ¿Qué es un etiqueta en Git?

Es una marca (o etiqueta) en un commit en particular. Esto nos será útil a la hora de señalar diferentes versiones de un mismo proyecto.

• ¿Cómo crear una etiqueta en Git?

Se creara ejecutando el comando git tag -a v1.0 -m “Version 1.0”

• ¿Cómo enviar una etiqueta a GitHub?

Una vez creada la etiqueta con el comando de ejemplo en la pregunta anterior, se enviara a git hub ejecutando git push origin v1.0

• ¿Qué es un historial de Git?

Es la lista de los commits que hemos realizado en un proyecto.

• ¿Cómo ver el historial de Git?

Lo podemos verificar ejecutando el comando git log.

• ¿Cómo buscar en el historial de Git?

Dentro del historial en general puedo buscar un commit en particular ejecutando el comando git log –grep=”nombre_del_commit”

• ¿Cómo borrar el historial de Git?

Lo hacemos ejecutando el comando rm -fr .git

• ¿Qué es un repositorio privado en GitHub?

Es un repositorio el cual puedo ver solo yo, o quienes tengan permiso(invitados).

• ¿Cómo crear un repositorio privado en GitHub?

Al momento de crear un repositorio normalmente, tendremos un casillero donde seleccionaremos Private.

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

Al posicionarnos sobre el repositorio privado que queremos compartir, nos dirigimos a settings(ajustes), ingresamos a Manage Access, y luego Invite a collaborater.

• ¿Qué es un repositorio público en GitHub?

Es un repositorio visible para todos en GitHub.

• ¿Cómo crear un repositorio público en GitHub?

Al momento de crear un repositorio (New Repository), debemos seleccionar Public.

• ¿Cómo compartir un repositorio público en GitHub?

Se comparte directamente con el enlace del repositorio.

2) Realizar la siguiente actividad:

• Crear un repositorio.
o Dale un nombre al repositorio.
o Elije el repositorio sea público.
o Inicializa el repositorio con un archivo.

• Agregando un Archivo
o Crea un archivo simple, por ejemplo, "mi-archivo.txt".
o Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.
o Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).

• Creando Branchs
o Crear una Branch
o Realizar cambios o agregar un archivo
o Subir la Branch

REPOSITORIO EJERCICIO 2:

https://github.com/Favio10/Repositorio_prueba_UTN

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.

REPOSITORIO EJERCICIO 3 :

https://github.com/Favio10/conflict-exercise