Alumno: Juarez Cristian

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.

1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

• ¿Qué es GitHub?

GitHub es un repositorio remoto es decir en la nube en la cual podemos compartir nuestros proyectos y trabajar con otros repositorios, clonarlos, compartir, etc.

• ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en GitHub, se debe iniciar sesión en la cuenta, luego tocar el botón de NEW, luego se debe elegir el nombre del repositorio, luego si lo decide se puede colocar una descripción que es opcional, luego se elige si será público o privado, y puedes optar por inicializarlo con un archivo README.md. y por ultimo tocar el botón de créate repository.

• ¿Cómo crear una rama en Git?

Para crear una rama en Git, se utiliza el comando: git branch rama1

• ¿Cómo cambiar a una rama en Git?

Para cambiar de rama se usa el comando: Git checkout rama2

• ¿Cómo fusionar ramas en Git?

Para fusionar ramas se utiliza el comando: Git merge rama 2

• ¿Cómo crear un commit en Git?

Para crear un commit en git se utiliza el comando: Git commit -m “primer commit”

• ¿Cómo enviar un commit a GitHub?

Para enviar un commit a github se utiliza el comando: Git push -u origin main

• ¿Qué es un repositorio remoto?

Un repositorio remoto es un lugar en la nube o servidor en donde nosotros podemos subir una copia de nuestros repositorios y trabajar con ellos o copartirlos en cualquier parte.

• ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto se utiliza el comando: git remote add

• ¿Cómo empujar cambios a un repositorio remoto?

Para empujar o hacer un push de un repositorio se utiliza el comando: Git push origin main

• ¿Cómo tirar de cambios de un repositorio remoto?

Para copiar cambios de un repositorio remoto a local se utiliza el comando: Git pull origin main

• ¿Qué es un fork de repositorio?

Un fork o bifurcación es copiar un repositorio de algún usuario en tu cuenta de GitHub de manera independiente.

• ¿Cómo crear un fork de un repositorio?

Para crear un fork de un repositorio, en GitHub se debe buscar el repositorio al cual le queremos realizar el fork, hacer clich en el botón fork , y el programa creara una coia local del repositorio.

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

Para enviar una solicitud de pull request desde GitHub, primero seleccionamos la rama que deseamos fusionar con la rama principal y hacemos clic en el botón “New Pull Request”.,

luego se completa la información requerida, se coloca el título una descripción del Pull Request, revisamos los cambios propuestos y por ultimo hacmos click en create pull request.

• ¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción o pull request en GitHub, primero se debe revisar los cambios propuestos, una vez revisados se puede aprobar la solicitud y luego fusionarla con la rama principal del repositorio. En el caso de que haya conflictos de fusión, primero se deben resolver antes de fusionar

• ¿Qué es un etiqueta en Git?

Una etiqueta o tag es una referencia que apunta a un punto específico en el historial de un repositorio. Se utiliza comúnmente para marcar versiones de software, como "v1.0" o "v2.1.2", pero también se puede usar para cualquier otro punto relevante en la historia del proyecto.

• ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta en Git se utiliza el comando: git tag “nombre de la version”

• ¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta o tag al repositorio remoto se utiliza el comando git push origin “nombre de la etiqueta”

• ¿Qué es un historial de Git?

El historial en Git es una representación de todas las modificaciones realizadas a un repositorio a lo largo del tiempo. Este historial se guarda a través de los commits que documentan el estado del proyecto en diferentes momentos.

• ¿Cómo ver el historial de Git?

Para ver el historial de commits en Git, se utiliza el comando git log

• ¿Cómo buscar en el historial de Git?

Para buscar en el historial de commits en Git, se puede usar el comando git log junto con diferentes opciones para filtrar y personalizar la salida. Por ejemplo, se puede puede usar git log --grep="palabra_a_buscar"

• ¿Cómo borrar el historial de Git?

Para borrar se utiliza el comando git rebase

• ¿Qué es un repositorio privado en GitHub?

En GitHub, un repositorio privado es un lugar donde se almacena código, archivos y el historial de revisiones el cual es solo es accesible para el propietario y las personas con las el usuario quiere compartir el acceso

• ¿Cómo crear un repositorio privado en GitHub?

Para hacer un repositorio de GitHub privado, se debecrear el repositorio y seleccionar la opción "Privado" en lugar de "Público". Luego, si lo desea se puede invitar a colaboradores y asignarles roles para controlar el acceso al repositorio.

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

1. Solicite el nombre de usuario de la persona que está invitando como colaboradora. Si
2. En el nombre del repositorio, haz clic en **Configuración**
3. En la sección "Acceso" de la barra lateral, haz clic en **Colaboradores**.
4. Haz clic en **Agregar personas**.
5. Buscar el nombre de la persona que deseas invitar dentro del campo de búsqueda.
6. Haga clic en **Agregar NOMBRE al REPOSITORIO**.
7. El usuario recibirá un correo electrónico invitándolo al repositorio. Una vez que acepte la invitación, tendrá acceso de colaborador a tu repositorio.

• ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un almacén de código, archivos y el historial de versiones que es accesible públicamente para cualquier persona en internet. Esto significa que cualquier persona puede ver, descargar y, en algunos casos, colaborar en el código sin necesidad de una invitación o permiso especial

• ¿Cómo crear un repositorio público en GitHub?

Para crear un repositorio público en GitHub, se debe inicia sesión en GitHub, haz clic en el signo más (+) en la esquina superior derecha y selecciona "Nuevo repositorio". Luego, ingresa el nombre del repositorio, la descripción (opcional), selecciona la opción "Público" para la visibilidad y crea el repositorio

• ¿Cómo compartir un repositorio público en GitHub?

Para compartir un repositorio en GitHub, puedes invitar colaboradores a tu repositorio personal o crear un repositorio dentro de una organización. También puedes compartir un repositorio privado mediante una URL, aunque esto puede implicar la generación de una clave y una URL de despliegue

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

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.