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 comunidad donde podemos compartir nuestros repositorios y ver los de otras personas. También sirve para nuestro repositorio disponible en la nube.

- ¿Cómo crear un repositorio en GitHub?

  Primero debemos crear una cuenta. Localizar el botón “New repository”, colocar un nombre al repositorio (es una buena práctica ponerle el mismo nombre que el repositorio local), opcionalmente podemos poner una descripción. Luego hacer clic en “Create repository”.

- ¿Cómo crear una rama en Git?

  Para crear una rama en Git, usamos el comando “git branch nombreDeLaRama”.

- ¿Cómo cambiar a una rama en Git?

  Utilizando el comando “git checkout nombreDeLaRama”.

- ¿Cómo fusionar ramas en Git?

  Estando dentro de la rama a la que queremos integrar los cambios, tipeamos “git merge *nombreDeLaRamaAIntegrar*”

- ¿Cómo crear un commit en Git?

  Primero, debemos preparar el o los archivos con “git add .” o “git add *nombreDeUnArchivoEspecifico”*

  Utilizando el comando “git commit -m *“Comentario que describe el/los cambios hechos al proyecto”* “.

- ¿Cómo enviar un commit a GitHub?

  Con el comando “git push origin *nombreDeLaRama*“.

- ¿Qué es un repositorio remoto?

  Es un repositorio alojado en la nube, el cual puede ser público o privado.

- ¿Cómo agregar un repositorio remoto a Git?

  Con el comando git remote set-url *URLdelRepositorioRemoto*

- ¿Cómo empujar cambios a un repositorio remoto?

  Primero debemos verificar el estado de los archivos que modificamos con “git status”.

  Luego con “git add .” preparamos los cambios.

  Después de esto creamos un registro de cambios con “git commit -m *“Comentario describiendo los cambios”* ”.

  Por último empujamos los cambios al repositorio remoto con “git push *nombreDeLaRama*”.

- ¿Cómo tirar de cambios de un repositorio remoto?

  Utilizando el comando “git pull *nombreDeLaRama*”.

  Se recomienda usarlo antes de empezar a trabajar con un código para evitar conflictos.

- ¿Qué es un fork de repositorio?

  Es una copia de un repositorio creada en una cuenta diferente, que permite realizar modificaciones al mismo sin afectar al original.

A diferencia del clonado, que crea una copia local, el fork genera una copia en la cuenta del usuario.

- ¿Cómo crear un fork de un repositorio?

  Dentro del repositorio del que se desea realizar un fork, presionamos el botón Fork, revisamos que los datos sean correctos y luego hacemos clic en “Create fork”.

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

  Si no se tiene permiso de escritura en el repositorio objetivo, deberemos hacer un fork del mismo.

  Luego clonamos localmente el fork creado con “git clone https://github.com/usuario/repositorio.git”

  A continuación, creamos una nueva rama para la modificación que haremos: “git checkout -b mi-mod” (Se crea una nueva rama y se cambia a ella).

  Hacemos las modificaciones en el código y preparamos y hacemos el commit con “git add .”, “git commit -m “*Descripción de mi modificación*” “.

  Subimos los cambios a nuestro fork “git push origin mi-mod”.

  Por último, ir a la página “Pull Requests” del repositorio original, hacer clic en “New Pull Request”.

Seleccionar:

  - **Base repository**: Repositorio original (usuario/repositorio).
  - **Base branch**: Rama donde quieres fusionar (ej: main).
  - **Head repository**: Tu Fork (tu-usuario/repositorio).
  - **Compare branch**: Tu rama (mi-mod).

Revisar los cambios y hacer clic en “Create Pull Request”. Luego de esto, completar los datos de la Pull Request (título, descripción, etc).

- ¿Cómo aceptar una solicitud de extracción?

  Para hacerlo de manera local, primero debemos sincronizar el repositorio local con “git fetch origin” y “git checkout main”.

  Luego “mergeamos” la rama de la Pull Request: “git merge origin/nombreDeLaRamaDeLaPullRequest.”

Por último subimos los cambios con “git push origin main”.

- ¿Qué es un etiqueta en Git?

  Las etiquetas son referencias que apuntan a puntos concretos en el historial de Git. Se usa para capturar un punto en el historial que se utiliza para una publicación de versión marcada (por ejemplo, v1.0.1).

  Una etiqueta es como una rama que no cambia, o sea que tras crearse, no tienen más historial de confirmaciones.

- ¿Cómo crear una etiqueta en Git?

  Se utiliza el comando “git tag *nombreDeLaEtiqueta*”. La etiqueta creada puede ser *ligera* o *anotada*.

  Las etiquetas *ligeras* son básicamente “marcadores” de una confirmación, son sólo un nombre y un puntero a una confirmación, útiles para crear enlaces rápidos. Ej: *git tag v1.4.* Para crear una etiqueta rápida basta con omitir los parámetros *-a, -s* o *-m*.

  Las etiquetas *anotadas* almacenan metadatos adicionales como el nombre de la persona que etiqueta, su email y fecha (datos importantes para una publicación pública). Se utiliza el comando *git tag -a v1.4 -m “Mi versión 1.4”*. Al igual que en un commit, se puede dejar una descripción con el parámetro -m.

  Si se omite el parámetro *-m*, se abrirá un editor de texto a fin de agregar metadatos.

- ¿Cómo enviar una etiqueta a GitHub?

  Se debe especificar la etiqueta junto con el comando *git push*. Ej: *git push v1.4*

- ¿Qué es un historial de Git?

  Es un registro de las confirmaciones de un repositorio, el cual se almacena como un gráfico de instantáneas.

  El historial de Git es importante para mantener la transparencia del proyecto, depurar errores y colaborar.

- ¿Cómo ver el historial de Git?

  Para ver el historial de commits de una rama, se utiliza *git log nombreDeLaRama.*

- ¿Cómo buscar en el historial de Git?

  Se utiliza el comando *git log* y se pueden agregar diferentes parámetros para comparar ramas, ver el log de commits graficado, ver commits donde un archivo especificado cambió.

- ¿Cómo borrar el historial de Git?

  Primero actualizamos el repositorio local con *git pull.*

  Luego, dentro del directorio local, creamos una nueva rama con *git checkout –orphan nuevaRama.*

  Después, agregamos todos los archivos a la rama nueva con *git add -A.*

  Realizamos un commit de los cambios: *git commit -am "Descripción de lo que estamos por subir".*

  Eliminamos la rama *main* o *master* con *git branch -D master.*

  Ahora renombramos la rama nueva a *main* o *master: git branch -m master.*

  Por último, pusheamos todo: *git push -f origin master.*

  Con esto, en la rama main del repositorio, no habrá registros de los commits previos.

- ¿Qué es un repositorio privado en GitHub?

  Un repositorio privado es un repositorio al que sólo tienen acceso el autor del mismo y los usuarios que él autorice.

- ¿Cómo crear un repositorio privado en GitHub?

  Al crear un repositorio nuevo en GitHub, marcar la opción *Private*. Luego esta opción se puede cambiar en *Settings -> Danger Zone -> Change repository visibility*.

  Desde la línea de comandos: *gh repo create nombreDelProyecto --private*

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

  Ir a *Settings -> Collaborators -> Add people* y agregar los nombres de usuario.

- ¿Qué es un repositorio público en GitHub?

  Es un proyecto de código abierto visible para cualquier persona.

- ¿Cómo crear un repositorio público en GitHub?

  Al crear un repositorio en GitHub, marcar la opción *Public*.

- ¿Cómo compartir un repositorio público en GitHub?

  Compartir la URL, la cual tiene el formato *https://github.com/tu-usuario/nombre-del-repo*

2) Realizar la siguiente actividad:

LINK A MI REPOSITORIO:

https://github.com/nicolima200/Ejercicio2Tp2.git

- 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:

LINK A MI REPOSITORIO:

https://github.com/nicolima200/Ejercicio3Tp2.git

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.