TRABAJO PRACTICO N° 2 (Git y GitHub)

PROGRAMACIÓN I

TUPaD-UTN

ACTIVIDADES:

1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada

(Desarrollar las respuestas):

• ¿Qué es GitHub?

Es una plataforma en línea que permite almacenar proyectos usando el sistema de control de versiones Git, es uno de los más populares, almacenando los repositorios en la nube. Este espacio remoto tiene herramientas para organizar, discutir, revisar y aprobar cambios propuesto por desarrolladores que realizan trabajos de modo privado o público.

• ¿Cómo crear un repositorio en GitHub?

Debemos crear una cuenta con usuario y contraseña en GitHub.com y una vez en el usuario de GitHub debemos iniciar un nuevo repositorio (hacer click en el simbolo “+” que se ubica en la parte superior derecha, al lado de la foto con menu desplegable del perfil) y a continuación rellenar los campos: nombre de repositorio, una descripcion (opcional), tipo de repositorio (público o privado) y de ese modo tendremos el repositorio creado. A posterior, solo queda subir la información que se encuentra en nuestra maquina local a los servidores de GitHub.

• ¿Cómo crear una rama en Git?

En primera medida debemos tener creado un proyecto alojado en el repositorio local y luego abrir la terminal o consola y debemos utilizar el comado “git init” para crear un repositorio locar por unica y a posterior con el comando “git branch <nombre de la rama>” se obtendra la creación de una rama dentro de Git.

• ¿Cómo cambiar a una rama en Git?

Para poder cambiar de ramas en Git, debemos utilizar el comando “git checkout <nombre de la rama> a la cual preferimos cambiar, destacar que para eso deben haber varias ramas creadas en primera instancia.

• ¿Cómo fusionar ramas en Git?

Fusionar ramas en Git se logra utilizando el comando “git merge <rama>”, el cual de este modo se obtendra la fusión de dos ramas distintas con información distintas, con el fin de que la información de ambas ramas se complemente y unifiquen su contenido.

• ¿Cómo crear un commit en Git?

Par poder crear un “commit” en Git, debemos preparar cambios en nuestros

archivos o proyectos, de lo contrario no se podra realizar el “commit” en la terminal o consola de Git. Es decir, debe haber existencia de cambios u/o modificaciones en nuestro archivo que se trabajó o que se este trabajando, ya que el comando “commit” significa que realiza una instantanea de los cambios realizados, captando el estado actualizado del proyecto. Por cada “commit” realizado podras observar el arcchivo en sus diferentes etapas de modificación. Es menester destacar que antes de realizar el “commit” se debe emplear el comando “git add .” para identificar los cambios realizados en el archivo, que a posterior seran guardados con el “commit”.

• ¿Cómo enviar un commit a GitHub?

En este caso para enviar un “commit” a GitHub se debe utilizar el comando “git push –u origin main” (en caso de que sea la primera vez que haces un “push”) y si con anteriorida ya se hicieron envios de commit a GitHub solo debes utilizar el comando “git push”

• ¿Qué es un repositorio remoto?

Un repositorio remoto es un sitio que aloja los proyectos atraves de repositorios creados por los desarrolladores, utilizando redes privadas (Universidades o empresas) o redes publicas (Internet) . En el repositorio remoto puedes observar proyectos de la comunidad de desarrolladores.

• ¿Cómo agregar un repositorio remoto a Git?

Los pasos para agregar un repositorio remoto a Git son los siguientes:

- Luego de loguearse en el repositorio remoto, copias la url (HTTPS o SSH).

- En la consola de Git debes pegar la URL copiada + Enter (esto se realiza por unica vez para tener asociado a Git nuestro repositorio remoto, para futuros cambios).

• ¿Cómo empujar cambios a un repositorio remoto?

Para EMPUJAR cambios a un repositorio remoto, en la consola o terminal se debe utilizar el comando “push”.

• ¿Cómo tirar de cambios de un repositorio remoto?

Para TIRAR de cambios de un repositorio remoto, en la consola o terminal se debe utilizar el comando “pull”. Antes de hacer el “pull” debes corroborar en la rama que te encuentras.

• ¿Qué es un fork de repositorio?

Un “fork**”** de un repositorio es una copia personal de un repositorio que se encuentra en una plataforma de control de versiones.

• ¿Cómo crear un fork de un repositorio?

Para crear un “Fork” de repositorio realizar una copia completa del repositorio de otro desarrollador desde GitHub; has clic en botón “Fork” ubicado en el repositorio creado por el usuario que estas visitando. A posterior debes clonar el repositorio

con el comando “git clone <url>” en la consola de Git.

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

El “pull request” se realiza para proponer cambios al proyecto original, y se realiza desde GitHub >>> pestaña “pull request” >>> new pull request >>>completar la solicitud de extracción. Asegúrate de que la base (base) sea la rama del repositorio original a la que deseas enviar tus cambios (por lo general, `main`), y que la comparación (compare) sea tu rama con los cambios.

• ¿Cómo aceptar una solicitud de extracción?

Ir a la pestaña de "Pull Requests" en el repositorio. Revisar los cambios propuestos y hacer clic en "Merge" para aceptar la solicitud.

• ¿Qué es una etiqueta en Git?

Una etiqueta (tag) es una referencia a un punto específico en la historia de un repositorio, generalmente utilizada para marcar versiones.

• ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta debemos ingresar el comando “git tag” seguido

de la versión que queremos utilizar.

• ¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta debemos utilizar el comando “git push -tags”.

• ¿Qué es un historial de Git?

El historial es donde se registran todos los cambios realizados en el

repositorio.

•¿Cómo ver el historial de Git?

Para ver el historial debemos escribir el comando “git reflog”.

•¿Cómo buscar en el historial de Git?

Para buscar en el historial podemos, por ejemplo, utilizar el comando

“git reflog show”

•¿Cómo borrar el historial de Git?

Podemos utilizar el comando “git reset --hard".

•¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es donde esta almacenado un trabajo

forma que nadie pueda verlo.

•¿Cómo crear un repositorio privado en GitHub?

Cuando queremos crear nuestro repositorio, debemos clickear en la opción “Private” debajo de la descripción.

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

Para invitar a una persona a nuestro repositorio privado debemos primero enviar nuestra invitación haciendo click en “Settings” y luego seleccionar en “Collaborators”. Por último, ya podemos agregar a nuestros compañeros yendo al botón “Add people”.

•¿Qué es un repositorio público en GitHub?

Un repositorio publico es el lugar donde este alojado nuestro proyecto y puede ser visto por terceros.

•¿Cómo crear un repositorio público en GitHub?

Cuando queremos crear nuestro repositorio, debemos clickear en la opción “Public” debajo de la descripción.

•¿Cómo compartir un repositorio público en GitHub?

Para invitar a una persona a nuestro repositorio público debemos primero enviar nuestra invitación haciendo click en “Settings” y luego seleccionar en “Collaborators”. Por último, ya podemos agregar a nuestros compañeros yendo al botón “Add people”.

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.