Práctico 2: Git y GitHub

Actividades

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

- **¿Qué es GitHub?**

GitHub es una plataforma en línea que se usa para guardar y trabajar con proyectos que usan Git. Sirve para tener un control de versiones, colaborar con otras personas en un mismo proyecto y también para subir código y tenerlo guardado en la nube. Es muy usada por programadores para trabajar en equipo y llevar un historial de los cambios que se hacen en los archivos.

- **¿Cómo crear un repositorio en GitHub?**

Para crear un repositorio en GitHub, primero hay que iniciar sesión en la cuenta. Después, se hace clic en el botón que dice "New" o "Nuevo repositorio". Ahí se completa el nombre del repositorio, se puede agregar una descripción (opcional), elegir si va a ser público o privado, y también se puede tildar la opción de agregar un README. Por último, se hace clic en "Create repository".

- **¿Cómo crear una rama en Git?**

Para crear una rama en Git, se usa el comando:

git branch *nombre-de-la-rama*

Ese comando crea la rama, pero no cambia a ella. Para cambiarse a esa rama, hay que usar:

git checkout *nombre-de-la-rama*

O también se puede hacer todo junto con:

git checkout -b *nombre-de-la-rama*

Eso crea la rama y te mueve directamente a ella. Las ramas sirven para trabajar en cosas nuevas sin afectar el código principal.

- **¿Cómo cambiar a una rama en Git?**

Para cambiar a una rama en Git, se usa el comando:

git checkout *nombre-de-la-rama*

- **¿Cómo fusionar ramas en Git?**

Para fusionar ramas en Git, primero hay que estar en la rama donde querés traer los cambios. Por ejemplo, si estás en main y querés fusionar otra rama (como nueva-funcion), hacés lo siguiente:

git checkout main

git merge *nueva-funcion*

• ¿Cómo crear un commit en Git?

Para crear un commit en Git, primero hay que agregar los archivos que se quieren guardar con este comando:

git add *nombre-del-archivo*

O si querés agregar todos los archivos modificados:

git add .

Después, se crea el commit con:

git commit **-m** "*Mensaje que describe los cambios*"

El mensaje debería explicar brevemente qué se hizo, así queda todo bien registrado en el historial del proyecto.

• ¿Cómo enviar un commit a GitHub?

Para enviar un commit a GitHub, primero tenés que haber hecho el commit localmente (con git commit). Después, se usa este comando para subirlo:

git push origin *nombre-de-la-rama*

Por ejemplo, si estás trabajando en la rama main, sería:

git push origin *main*

Eso sube los cambios al repositorio en GitHub, y quedan guardados ahí para que otros también los puedan ver o seguir trabajando.

• ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión del repositorio que está guardada en internet o en una red, como por ejemplo en GitHub. Sirve para tener una copia del proyecto fuera de tu computadora, compartirlo con otras personas y colaborar en equipo. Con Git se puede conectar el repositorio local (el que tenés en tu compu) con el remoto para subir o bajar cambios.

• ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git, se usa el siguiente comando:

git remote add origin *URL-del-repositorio*

Por ejemplo:

git remote add origin *https://github.com/usuario/nombre-del-repo.git*

Ese comando le dice a Git dónde está el repositorio remoto (como el de GitHub) y le pone el nombre "origin" por defecto. Una vez agregado, ya se pueden usar comandos como git push o git pull para sincronizar los cambios.

• ¿Cómo empujar cambios a un repositorio remoto?

Para empujar los cambios hechos en el repositorio local hacia uno remoto (como GitHub), se usa el comando:

git push origin *nombre-de-la-rama*

Esto asegura que los commits locales se reflejen también en el repositorio remoto, manteniendo todo actualizado y sincronizado.

• ¿Cómo tirar de cambios de un repositorio remoto?

Para traer (o "tirar") los últimos cambios desde el repositorio remoto al local, se usa:

git pull origin *nombre-de-la-rama*

Por ejemplo:

git pull origin *main*

Este comando actualiza tu repositorio local con los cambios más recientes que haya en GitHub (u otro remoto), así trabajás siempre con la versión más actualizada del proyecto.

• ¿Qué es un fork de repositorio?

Un fork es una copia de un repositorio que se crea en tu cuenta de GitHub. Sirve para que puedas modificar un proyecto sin afectar el original. Es muy usado cuando querés colaborar con un proyecto ajeno: hacés un fork, trabajás en tu copia y después podés mandar tus cambios al repositorio original con un pull request.

• ¿Cómo crear un fork de un repositorio?

Para crear un fork, primero hay que entrar al repositorio que querés copiar en GitHub. Una vez ahí, hacés clic en el botón que dice "Fork" (generalmente está arriba a la derecha). GitHub va a crear una copia del repositorio en tu cuenta, y desde ahí ya podés hacer cambios sin afectar el original.

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

Después de hacer cambios en tu fork y subirlos a GitHub, hay que ir al repositorio original. Ahí, GitHub te va a mostrar un botón que dice **"Compare & pull request"**. Hacés click ahí, escribís un mensaje que explique qué cambiaste, y después seleccionas **"Create pull request"**. Eso envía la solicitud para que los dueños del proyecto revisen tus cambios y, si están de acuerdo, los fusionen con el repositorio original.

• ¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción (pull request), tenés que ser colaborador o tener permisos en el repositorio original. Cuando alguien manda un pull request, vas a ver una notificación o podés entrar a la pestaña **"Pull requests"**. Desde ahí, revisás los cambios, y si todo está bien, hacés clic en **"Merge pull request"** y luego en **"Confirm merge"**. Eso agrega los cambios al proyecto principal.

• ¿Qué es una etiqueta en Git?

Una etiqueta (o *tag*) en Git se usa para marcar un punto específico en la historia del proyecto, como una versión importante. Por ejemplo, cuando se lanza una nueva versión de una app, se suele crear una etiqueta para identificar ese momento. Es como poner un marcador que dice "acá salió la versión 1.0".

• ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta en Git, se usa el comando:

git tag *nombre-de-la-etiqueta*

Por ejemplo:

git tag *v1.0*

• ¿Cómo enviar una etiqueta a GitHub?

Una vez que creaste una etiqueta con git tag, para enviarla a GitHub hay que hacer lo siguiente:

git push origin *nombre-de-la-etiqueta*

Por ejemplo:

git push *origin v1.0*

• ¿Qué es un historial de Git?

El historial de Git es el registro de todos los cambios que se hicieron en un proyecto. Guarda cada commit con su autor, fecha y mensaje, lo que permite ver cómo fue evolucionando el código.

• ¿Cómo ver el historial de Git?

Para ver el historial de Git, se usa el comando:

git **log**

• ¿Cómo buscar en el historial de Git?

Para buscar algo específico en el historial de Git, se puede usar:

git **log** **--grep**=*"palabra clave"*

Eso busca en los mensajes de los commits. Por ejemplo:

git **log** **--grep**="*login*"

• ¿Cómo borrar el historial de Git?

Borrar el historial de Git no es algo común, pero se puede hacer si querés empezar desde cero. Una forma de hacerlo es creando un nuevo *branch* huérfano (sin historial) y forzar el push:

git checkout **--orphan** *nueva-rama*

git add .

git commit **-m** "*Nuevo comienzo*"

git push **-f** origin *nueva-rama*

Después se puede borrar la rama anterior en GitHub si querés que sólo quede la nueva. (*Esto borra el historial anterior y puede afectar a otros que trabajen en el proyecto.)*

• ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un proyecto que solo pueden ver y acceder las personas que autorices. A diferencia de los repositorios públicos, que cualquiera puede ver, los privados sirven para trabajar en equipo de forma cerrada o para guardar código que no querés compartir con todo el mundo. Podés cambiar un repositorio de público a privado (o al revés) desde la configuración del repositorio.

• ¿Cómo crear un repositorio privado en GitHub?
Primero, iniciar sesión en GitHub y hacer clic en el botón **"New"** para crear un nuevo repositorio. Completar el nombre, la descripción (opcional), y luego seleccionar la opción **"Private"**. Y finalmente hacer click en **"Create repository"**.

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

Entrás al repositorio privado y vas a la pestaña **"Settings"**. Después, en el menú lateral elegís **"Collaborators"** o **"Manage access"** (según cómo esté organizado). Ahí hacés click en **"Invite a collaborator"**, escribís el nombre de usuario de la persona y confirmás la invitación.

Cuando acepte, va a poder acceder al repo y trabajar con vos.

• ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un proyecto que cualquier persona puede ver, clonar y hasta proponer cambios (mediante pull requests). Es ideal para compartir código abierto, colaborar con otros o mostrar tus trabajos al mundo.

• ¿Cómo crear un repositorio público en GitHub?

Entrás a GitHub y hacés clic en el botón **"New"** para crear un nuevo repositorio. Completás el nombre, la descripción (opcional), y seleccionas la opción **"Public"**. Después hacés clic en **"Create repository"** y listo, tu código va a estar disponible para que cualquiera lo vea o lo use.

• ¿Cómo compartir un repositorio público en GitHub?

Entrás al repositorio y copiás la URL desde la barra del navegador o hacés clic en el botón "Code" y copiás el enlace HTTPS. Después lo pegás donde quieras (chat, mail, redes, etc.). Como el repositorio es público, cualquiera con ese link puede acceder y verlo.