Trabajo práctico 2: Git y GitHub

Actividad 1:

1. GitHub es una plataforma en línea que permite gestionar proyectos de software utilizando el sistema de control de versiones Git. Funciona como un repositorio donde los desarrolladores pueden almacenar, compartir y colaborar en código de manera organizada y segura.
2. Para crear un repositorio en GitHub, primero es necesario contar con una cuenta en la plataforma. Una vez dentro, se puede acceder a la opción de creación de un nuevo repositorio, donde se solicita asignar un nombre al proyecto, una descripción opcional, y elegir si será público o privado, dependiendo del nivel de acceso que se desee permitir. También existe la posibilidad de inicializar el repositorio con un archivo README, lo cual es recomendable ya que facilita la organización del proyecto desde el inicio. Al finalizar este proceso, el repositorio queda creado y disponible para comenzar a trabajar en él, ya sea directamente desde la web o vinculándolo con un entorno de desarrollo local mediante Git.
3. Crear una rama en Git es una práctica fundamental para el desarrollo organizado de software, ya que permite trabajar en nuevas funcionalidades o correcciones sin afectar directamente el código principal del proyecto. Una rama es básicamente una versión paralela del repositorio, donde se pueden realizar cambios de forma aislada. Para crearla, se utiliza el comando correspondiente desde la línea de comandos o terminal, y una vez generada, se puede cambiar a esa nueva rama para empezar a trabajar sobre ella.
4. Cambiar a una rama en Git es un paso necesario cuando se quiere trabajar sobre una versión específica del proyecto. Esto se realiza comúnmente después de haber creado una nueva rama o cuando se desea volver a una ya existente. Git permite hacer este cambio de manera simple mediante un comando en la terminal, lo que actualiza el entorno de trabajo con los archivos y el historial correspondiente a la rama seleccionada. Al cambiar de rama, el desarrollador puede enfocarse en una tarea puntual sin interferir con el desarrollo que ocurre en otras ramas.
5. Fusionar ramas en Git, también conocido como *merge*, es el proceso mediante el cual se integran los cambios realizados en una rama dentro de otra. Generalmente, esta acción se lleva a cabo cuando se termina de trabajar en una funcionalidad o corrección dentro de una rama secundaria, y se desea incorporar ese trabajo a la rama principal del proyecto. Para realizar la fusión, primero se debe posicionar el proyecto en la rama que recibirá los cambios y luego ejecutar el comando correspondiente para integrar el contenido de la otra rama.
6. Crear un commit en Git es uno de los pasos más importantes en el control de versiones, ya que permite guardar un conjunto de cambios realizados en los archivos del proyecto, junto con un mensaje descriptivo que explica qué se modificó. Este proceso es esencial para llevar un seguimiento ordenado del desarrollo, ya que cada commit representa un punto en la historia del proyecto al que se puede volver en caso de ser necesario. Para crear un commit, primero se deben seleccionar los archivos que se quieren guardar utilizando el comando de *add*, lo que los prepara para ser registrados. Luego, se utiliza el comando de *commit* junto con un mensaje que describa brevemente los cambios realizados.

7. Enviar un commit a GitHub es el proceso mediante el cual los cambios realizados en un repositorio local se suben al repositorio remoto en GitHub. Este paso es esencial para compartir el trabajo con otros colaboradores o para tener una copia de seguridad en la nube. Una vez que se ha realizado un commit en el repositorio local, el siguiente paso es "empujar" esos cambios al repositorio de GitHub. Para hacerlo, se utiliza el comando git push, que transfiere los commits desde el repositorio local hacia el repositorio remoto en la plataforma de GitHub.
8. Un repositorio remoto es una versión de un repositorio de Git que está almacenada en un servidor o plataforma en línea, como GitHub, en lugar de estar en la máquina local de un desarrollador. La principal ventaja de un repositorio remoto es que permite la colaboración entre varios desarrolladores, ya que todos pueden acceder al mismo repositorio desde distintas ubicaciones. A través de comandos como git push y git pull, los desarrolladores pueden sincronizar sus cambios con el repositorio remoto, enviando sus actualizaciones o descargando las modificaciones realizadas por otros.
9. Agregar un repositorio remoto a Git permite vincular un repositorio local con uno alojado en una plataforma en línea, como GitHub, para poder compartir y sincronizar cambios. Este proceso se realiza mediante el comando git remote add, que le indica a Git la ubicación del repositorio remoto y le asigna un nombre, generalmente "origin" (aunque se puede usar cualquier nombre). Al configurar un repositorio remoto, Git sabe a dónde enviar los cambios cuando se ejecute el comando git push, y también desde dónde descargar los cambios con el comando git pull.
10. Empujar cambios a un repositorio remoto en Git, conocido como *push*, es el proceso mediante el cual se suben los commits realizados en un repositorio local a un repositorio remoto, como GitHub. Este paso es necesario para compartir los avances con otros colaboradores o para guardar el trabajo en un servidor en la nube.
11. Tirar cambios de un repositorio remoto en Git, conocido como pull, es el proceso mediante el cual se descargan los cambios más recientes desde un repositorio remoto hacia tu repositorio local. Esto es útil cuando otros colaboradores han realizado modificaciones y las han subido al repositorio remoto, y tú necesitas mantener tu copia local actualizada con esos cambios.
12. Un fork de repositorio es una copia independiente de un repositorio original, que se crea generalmente en plataformas como GitHub. Este proceso permite que un desarrollador pueda realizar cambios en un proyecto sin afectar el repositorio original, lo cual es especialmente útil cuando se quiere colaborar en proyectos de código abierto. El fork actúa como una bifurcación del repositorio principal, permitiendo al usuario trabajar de forma aislada en nuevas funcionalidades, correcciones de errores o mejoras, sin modificar directamente el código de la fuente original.
13. Crear un *fork* de un repositorio es un proceso sencillo que permite crear una copia independiente de un proyecto en plataformas como GitHub. Esto es útil cuando se quiere colaborar en un proyecto de código abierto o experimentar con el código sin afectar el repositorio original.

Para crear un fork de un repositorio, primero debes acceder al repositorio que deseas copiar en GitHub. Una vez dentro de la página del repositorio, verás un botón en la parte superior derecha que dice "Fork". Al hacer clic en este botón, GitHub creará una copia del repositorio en tu cuenta, que podrás modificar y trabajar de manera independiente del repositorio original.

14. Enviar una solicitud de extracción, conocida como pull request, es el proceso mediante el cual un desarrollador propone cambios realizados en su propio repositorio para que sean revisados e integrados en el repositorio original. Este mecanismo es fundamental en proyectos colaborativos, ya que permite mantener un control sobre los aportes que se incorporan al código principal, promoviendo la revisión entre pares y la mejora continua del proyecto.

    Para enviar una *pull request*, primero es necesario haber realizado cambios en un repositorio, ya sea a través de un *fork* o trabajando en una rama distinta dentro del mismo proyecto.

15. Aceptar una solicitud de extracción en GitHub es el paso final para integrar los cambios propuestos por un colaborador al repositorio original. Este proceso lo realiza generalmente un mantenedor o administrador del repositorio, y es fundamental para asegurar que los cambios se revisen y validen antes de formar parte del código principal del proyecto.

    Para aceptar una pull request, primero se debe ingresar al repositorio y acceder a la sección de pull requests, donde se listan todas las solicitudes abiertas. Al seleccionar una de ellas, se puede revisar el resumen de los cambios realizados, los archivos modificados y los comentarios de otros colaboradores si los hubiera.

16. Una etiqueta en Git, conocida como tag, es una referencia que se utiliza para marcar puntos específicos en la historia del repositorio, generalmente para señalar versiones importantes del proyecto, como lanzamientos o versiones estables. A diferencia de una rama, una etiqueta no cambia con el tiempo ni se mueve; apunta a un commit fijo que representa un estado particular del código.

17. Crear una etiqueta en Git es una forma efectiva de marcar un punto específico en la historia del proyecto, generalmente asociado a una versión estable o a un hito importante. Este proceso permite identificar fácilmente el estado del código en un momento determinado, lo cual es especialmente útil en el control de versiones y en la publicación de software.

18. Enviar una etiqueta a GitHub es el proceso mediante el cual se sube una etiqueta creada localmente en un repositorio Git a su versión remota alojada en la plataforma, permitiendo que otros colaboradores también puedan acceder a ella. Esto es útil, por ejemplo, cuando se lanza una nueva versión del software y se quiere que esté disponible públicamente o para el equipo de desarrollo.

19. El historial de Git es el registro completo de todos los cambios realizados en un repositorio a lo largo del tiempo. Este historial se compone de *commits*, que son puntos de control que guardan el estado del proyecto en momentos específicos, junto con información sobre qué archivos se modificaron, quién hizo los cambios, cuándo se hicieron y un mensaje descriptivo que explica el propósito de cada cambio.

20. Ver el historial de Git es una forma fundamental de revisar y comprender cómo ha evolucionado un proyecto a lo largo del tiempo. Git ofrece varias maneras de acceder a este historial, siendo la más común el uso del comando git log desde la terminal. Este comando muestra una lista cronológica de los commits realizados, incluyendo información clave como el hash del commit, el autor, la fecha y el mensaje descriptivo asociado a cada cambio.

21. Buscar en el historial de Git es una herramienta muy útil para encontrar cambios específicos, identificar cuándo se modificó cierto fragmento de código o entender el contexto de alguna actualización en el proyecto. Git ofrece varios comandos y opciones que permiten explorar el historial de manera eficiente y precisa, especialmente en proyectos con muchos commits.

22. Borrar el historial de Git no es una operación común ni recomendada en la mayoría de los casos, ya que va en contra del principio central de Git: mantener un registro detallado y trazable de todos los cambios realizados en un proyecto. Sin embargo, en situaciones específicas —como cuando se quiere reiniciar un repositorio, eliminar información sensible o simplificar el historial antes de publicar un proyecto— es posible hacerlo, aunque con precaución.

    Una de las formas más directas de borrar el historial es creando una nueva rama huérfana (*orphan branch*), lo que genera un nuevo historial desde cero, sin conexión con los commits anteriores.

23. Un repositorio privado en GitHub es un espacio de trabajo que permite almacenar código y archivos de forma confidencial, es decir, que solo es accesible para el usuario que lo creó y para los colaboradores que este autorice explícitamente. A diferencia de los repositorios públicos, que pueden ser vistos y clonados por cualquier persona en Internet, los repositorios privados están protegidos y ocultos, lo que los hace ideales para proyectos personales, empresariales o en desarrollo que aún no están listos para ser compartidos con el público.

24. Crear un repositorio privado en GitHub es una excelente opción cuando deseas trabajar en un proyecto que no deseas que sea visible para el público. Este tipo de repositorios solo está accesible para el creador y los colaboradores a los que se les otorgue permisos explícitos. Para crear un repositorio privado, primero debes acceder a tu cuenta en GitHub y comenzar el proceso de creación de un nuevo repositorio. Durante la configuración, podrás asignarle un nombre y, opcionalmente, una descripción. Al seleccionar la opción de visibilidad, debes elegir la opción

"Private", lo que asegurará que el repositorio sea accesible solo para las personas que invites.

25. Para invitar a alguien a un repositorio privado, debes ser el propietario del repositorio o tener permisos de administrador. Desde la interfaz de GitHub, puedes ir a la página de tu repositorio privado, y allí, en la sección de configuración (Settings), encontrarás una opción llamada "Manage access" o "Administrar acceso". En esta sección, podrás invitar a colaboradores proporcionando sus nombres de usuario de GitHub.

    Al hacer clic en "Invite a collaborator", podrás ingresar el nombre de usuario o el correo electrónico de la persona que deseas invitar. Una vez que la persona acepte la invitación, podrá acceder al repositorio de acuerdo con los permisos que le hayas otorgado, como solo lectura, escritura o administración completa.

26. Un repositorio público en GitHub es un espacio de almacenamiento de código y archivos que está abierto y accesible para cualquier persona en Internet. Cualquier usuario puede ver, clonar, bifurcar y contribuir a un repositorio público sin necesidad de permisos especiales. Este tipo de repositorio es ideal para proyectos de código abierto, donde la comunidad puede colaborar, revisar el código y mejorar el proyecto de manera transparente y accesible.

27. Para crear un repositorio público, primero necesitas iniciar sesión en tu cuenta de GitHub. Luego, debes acceder a la página principal de tu cuenta y hacer clic en el botón **"+"** en la parte superior derecha de la pantalla, donde seleccionarás la opción **"Nuevo repositorio"**.

    Una vez en la página de creación del repositorio, deberás proporcionar un nombre para el repositorio y, si lo deseas, agregar una descripcion para explicar de qué trata el proyecto. A continuación, se te dará la opción de elegir la visibilidad del repositorio. Para crear un repositorio público, selecciona la opción **"Publico"**.

28. Una de las formas más simples es copiar la **URL** del repositorio desde la barra de direcciones de tu navegador mientras estás en la página principal del repositorio en GitHub. Puedes compartir este enlace a través de correos electrónicos, redes sociales, foros, o cualquier otra plataforma para invitar a otros a que vean o contribuyan a tu proyecto.