Trabajo Practico N°2: Git-GitHub
Nombre: *BRIÑOCCOLI Adrian Nahuel*

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

1.1) ¿Qué es GitHub?

Es una plataforma de desarrollo colaborativo basada en la web que utiliza el sistema de control de versiones Git. Es ampliamente utilizado por desarrolladores para alojar, revisar y7 colaborar en proyectos de software.

1.2) ¿Cómo crear un repositorio en GitHub?

Crear un repositorio en GitHub implica generar un espacio en la plataforma donde se puede almacenar y gestionar tu código de manera organizada. Primero, necesitas una cuenta en GitHub. Una vez adentro, puedes crear un nuevo repositorio dándole un nombre y eligiendo si será público o privado. GitHub te permite inicializar el repositorio con un archivo README, que es útil para describir el propósito del proyecto.
Si quieres trabajar en el repositorio desde tu computadora, puedes clonarlo utilizando Git. Esto descarga el contenido del repositorio en tu máquina, permitiéndote hacer cambios localmente. Luego, puedes agregar archivos, guardarlos en el historial de cambios con un commit y subirlos de nuevo a GitHub con un Push. Esto facilita el trabajo colaborativo y el control de versiones de tu proyecto.

1.3) ¿Cómo crear una rama en Git?

En Git, una rama (o **Branch**) es una versión separada del código donde puedes trabajar sin afectar la versión principal (**main o master**). Para crear una rama, primero necesitas estar dentro del repositorio de Git, puedes crear una nueva rama con el comando (***git branch nombre-de-la-rama)***.

1.4) ¿Cómo cambiar a una rama de Git?

Para moverte a la nueva rama hay que utilizar el comando (***git checkout nombre-dela-rama)***.

1.5) ¿Cómo fusionar ramas en Git?

Para fusionar **(merge)** ramas en git, primero debes asegurarte de estar en la rama donde quieres combinar los cambios. Normalmente, se fusiona una rama secundaria en la principal. Para fusionar la rama secundaria en la rama principal hay que utiliza el siguiente comando ***(git merge nombre-de-la-rama)***.

1.6) ¿Cómo crear un commit en Git?

Un commit en Git es como un punto de guardado en la evolución de un proyecto. Cada vez que realizas cambios en los archivos, puedes agruparlos en un commit para registrar el estado del código en ese momento. Esto te permite rastrear que modificaciones se hicieron y quien las realizo.
Para hacer un commit primero es necesario agregar los archivos al área de preparación, lo que indica Git que cambios se incluirán en ese registro. Luego se utiliza el comando ***(git commit –m "con un mensaje descriptivo que explique los cambios que se realizaron")***

1.7) ¿Cómo enviar un commit a GitHub?

Una vez hecho el commit, los cambios quedan almacenados localmente en el historial del repositorio. Sin embargo, si se trabaja con un repositorio remoto, es necesario utilizar el

siguiente comando ***(git push origin nombre-de-la-rama)*** para enviar los cambios y compartirlos con otros colaboradores.

1.8) ¿Qué es un repositorio remoto?

Un repositorio remoto en Git es una versión de tu repositorio que está almacenada en un servidor en línea. A diferencia del repositorio local, que solo existe en tu computadora, el remoto permite compartir tu código con otros desarrolladores y mantener una copia segura del proyecto.

1.9) ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git, primero necesitas un repositorio local. Luego, puedes vincularlo con un repositorio remoto alojado en la plataforma de GitHub. Una vez creado, la plataforma te proporcionara una URL que representa la ubicación remota del repositorio. En la terminal, debes ejecutar un comando para conectar tu repositorio local con el remoto, esto le indica a Git donde realizar los cambios.

1.10) ¿Cómo empujar cambios a un repositorio remoto?

Antes de empujar los cambios, es una buena opción obtener los últimos cambios des repositorio remoto para evitar conflictos, para eso utilizamos el comando ***(git pull origin nombre-de-la-rama)*** y para subir los cambios utilizar el comando ***(git push origin nombre-de-la-rama).***

1.11) ¿Cómo tirar de cambios de un repositorio remoto?

Para tirar los cambios del repositorio remoto hay que usar el comando ***(git pull)*** para descargar y fusionar los cambios del repositorio remoto con la rama local.

1.12) ¿Qué es un fork de repositorio?

Un fork de un repositorio es una copia independiente de un repositorio que se encuentra en otra cuenta dentro de GitHub, hacer un fork te permite obtener una versión completa de un proyecto sin afectar el repositorio original. El principal uso de un fork es contribuir a proyector de código abierto, se puede hacer cambios en la copia del repositorio.

1.13) ¿Cómo crear un fork de un repositorio?

Crear un ***fork*** de un repositorio en GitHub es un proceso sencillo que te permite hacer una copia independiente de un proyecto en tu propia cuenta. Para hacerlo, primero debes ir al repositorio original en GitHub y hacer clic en el botón (Fork), que se encuentra en la parte superior derecha de la página.

Al hacer esto, GitHub creará una copia exacta del repositorio en tu cuenta, esta copia es completamente independiente del original, lo que significa que puedes modificarla sin afectar el repositorio principal.

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

Para enviar una solicitud de extracción (pull request) a un repositorio en GitHub, primero necesitas haber realizado un fork del repositorio original y haber hecho cambios en tu propia cuenta, cuando ya están las modificaciones listas en tu repositorio, debes ir a la sección de “Pull Requests” en el repositorio original. Allí encontraras una opción para crear una nueva solicitud. GitHub te pedirá que selecciones la rama desde la cual deseas enviar los cambios y a que rama del repositorio deseas enviar los cambios y a que rama del repositorio original quieres fusionarlos.

Es importante agregar un mensaje claro y descriptivo que explique qué cambios has realizado y porque deben ser incorporados en el proyecto principal.

1.15) ¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción (pull request), primero debes tener permisos de escritura o ser el mantenedor del repositorio donde se hizo la solicitud.
Cuando alguien envía un pull request, puedes revisarlo en la pestaña "PULL REQUEST" del repositorio. Es recomendable revisar los cambios propuesto antes de aceptarlo, se puede hacer esto observando el resumen de modificaciones y comentando sobre las líneas específicas del código si es necesario. Si todo está en orden y no hay conflictos con la rama principal del proyecto, puedes hacer clic en el botón "MERGE PULL REQUEST" para fusionar los cambios en la rama correspondiente.

1.16) ¿Qué es una etiqueta en Git?

Git tiene la posibilidad de etiquetar puntos específicos del historial como importantes, esta función se usa típicamente para marcar versiones de lanzamiento.

1.17) ¿Cómo crear una etiqueta en Git?

Para crear etiquetas en Git, primero decides si será una etiqueta ligera o anotada, una ligera es simplemente un marcador de commit, mientras que una anotada almacena información adicional como el autor y un mensaje descriptivo.
Las etiquetas anotadas se guardan en la base de datos de Git como objetos enteros, tienen un checksum este contiene el nombre del etiquetador, correo electrónico y fecha, tienen un mensaje asociado. Generalmente se recomienda realizar etiquetas anotadas, de esta manera se tenga toda la información.

1.18) ¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta a GitHub, primero debes haberla creado en tu repositorio local, las etiquetas no se envían automáticamente cuando haces un push, por lo que debes enviarlas de manera explícita. Si deseas compartir una sola etiqueta, puedes enviarla de forma individual al repositorio remoto, si prefieres enviar todas las etiquetas que has creado en tu repositorio local, también puedes hacerlo con un comando (git push origin –tags) que las subes en conjunto.

1.19) ¿Qué es un historial de Git?

El historial de Git es el registro de todos los cambios realizado en un repositorio a lo largo del tiempo, cada vez que se crea un commit, Git almacena información detallada sobre lo que se modificó, quien hizo los cambios y en qué momento. Este historial te permite rastrear la evolución del proyecto, comparar versiones anteriores y, si es necesario, revertir cambios no deseados. Además el historial no solo guarda los commits, sino también refleja la creación y fusión de ramas, etiquetas y contribuciones.

1.20) ¿Cómo ver el historial de Git?

Para ver el historial de Git, existen varios comandos que permiten analizar los commits y los cambios realizados en un repositorio. El comando más común para ver el historial es (git log), que muestra una lista de commits con detalles como el identificador único, el autor, la fecha y el mensaje del commit. También se puede aplicar opciones para personalizar la visualización, como (git log –oneline), que muestra un resumen más compacto.

1.21) ¿Cómo buscar en el historial de Git?

Para buscar en el historial de Git, se puede usar diferentes opciones de comandos (git log), que permiten filtrar los commits según distintos criterios. Si se necesita encontrar un commit especifico por su mensaje, se utiliza el siguiente comando (git log –grep=”palabra clave”), lo que mostrará solo los commits que contengan esa palabra en su descripción.
Por autor: (git log –author=”nombre”)
Por fecha: (git log – since=”YYYY-MM-DD”)

1.22) ¿Cómo borrar el historial de Git?

Borrar el historial de Git puede significar diferentes cosas dependiendo de lo que realmente quieras lograr. Git no permite eliminar completamente el historial de manera simple, ya que está diseñado para preservar el registro de cambios. Sin embargo, hay maneras de "reiniciar" o eliminar el historial según la necesidad.

Si uno desea borrar el historial de commits pero mantener los archivos en su estado actual, puede reiniciar el repositorio y comenzar de nuevo sin perder el contenido.

Los comandos pueden ser:

- git reset: esto quita de stage todos los archivos y carpetas del proyecto.
- git reset nombre-archivo: quita del stage el archivo indicado.
- git reset nombre-carpeta: quita del stage todos los archivos de esa carpeta.
- git reset nombre-carpeta/nombre-archivo: quita ese archivo del stage, que a su vez está adentro de una carpeta.
- git reset nombre-carpeta/*.extensión: quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta stage.

1.23) ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un repositorio cuyo contenido no es accesible públicamente. Solo las personas a las que se les haya concedido permiso pueden verlo y contribuir en él. A diferencia de los repositorios públicos, que cualquier usuario de GitHub puede explorar y clonar, un repositorio privado está protegido y solo es visible para los colaboradores autorizados.

1.24) ¿Cómo crear un repositorio privado en GitHub?

Para crear un repositorio privado en GitHub, primero debes acceder a tu cuenta y dirigirte a la opción para crear un nuevo repositorio. Durante la configuración, puedes elegir entre un repositorio público o privado. Si seleccionas la opción privada, el código solo será accesible para ti y para las personas a las que concedas permisos.

Una vez creado, puedes gestionarlo de la misma manera que cualquier otro repositorio, pero con la ventaja de que no será visible para otros usuarios a menos que los invites. Además, puedes cambiar la configuración en cualquier momento si deseas hacerlo público más adelante. Esta opción es ideal para proyectos en desarrollo que requieren confidencialidad o control de acceso.

1.25) ¿Cómo invitar a alguien a un repositorio privado en GitHub?

Para invitar a alguien a un repositorio privado en GitHub, debes ser el propietario del repositorio o tener permisos de administración. Desde la configuración del repositorio, puedes acceder a la sección de "Colaboradores" o "Equipos" y agregar el nombre de usuario o correo electrónico de la persona que deseas invitar.

GitHub enviará una invitación a la persona, quien deberá aceptarla antes de obtener acceso al repositorio. Además, puedes asignarle diferentes niveles de permisos, como solo lectura o permisos de escritura para realizar cambios. Esta funcionalidad es útil para colaborar en proyectos privados sin hacer público el código.

1.26) ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio cuyo contenido es accesible para cualquier persona en internet. Cualquier usuario de GitHub puede ver, clonar y descargar el código sin necesidad de permisos especiales. Sin embargo, solo los colaboradores con acceso pueden realizar cambios en el repositorio.

Este tipo de repositorio es ideal para proyectos de código abierto, donde se busca que otros desarrolladores puedan contribuir, reportar problemas o hacer sugerencias. Además, GitHub

permite gestionar permisos para controlar quién puede realizar cambios, mientras que el historial de modificaciones queda registrado de manera transparente.

1.27) ¿Cómo crear un repositorio público en GitHub?

Para crear un repositorio público en GitHub, primero debes iniciar sesión en tu cuenta y acceder a la opción para crear un nuevo repositorio. Durante la configuración, GitHub te permite elegir entre un repositorio público o privado. Si seleccionas la opción pública, cualquier persona podrá ver y clonar el código, pero solo los colaboradores con permisos podrán modificarlo.
Una vez creado, el repositorio estará disponible para toda la comunidad de GitHub. Puedes agregar archivos, realizar *commits* y administrar ramas como en cualquier otro repositorio.

1.28) ¿Cómo compartir un repositorio público en GitHub?

Para compartir un repositorio público en GitHub, simplemente necesitas proporcionar la URL del repositorio. Cualquier persona con ese enlace podrá acceder al código, clonarlo y explorarlo sin necesidad de permisos especiales.
Si uno desea facilitar el acceso, puedes compartir el enlace en redes sociales, foros o documentación del proyecto. También puedes agregar colaboradores con permisos específicos si quieres que otras personas contribuyan directamente. Además, GitHub ofrece la opción de generar un enlace de invitación para contribuir, lo que permite a otros hacer *push* a ramas sin necesidad de ser añadidos como colaboradores permanentes.