Programacion – TUP

Profesor: Sebastian Bruselario

Alumno: Facundo Miguel Archiria

Trabajo Práctico N.º 2 (Trabajo Colaborativo): Git y GitHub

¿Qué es GitHub?

GitHub es como una red social para programadores, pero especializada en código. Es una plataforma en línea donde podemos subir nuestros proyectos de programación usando Git, un sistema que nos ayuda a gestionar las diferentes versiones de nuestro código. Pienso que su mayor valor está en que nos permite colaborar con otros, hacer revisiones, comentar cambios y mantener todo organizado de manera clara. Además, en GitHub podemos almacenar repositorios tanto públicos (que todos pueden ver) como privados (solo nosotros y quienes autoricemos).

¿Cómo crear un repositorio en GitHub?

Primero, ingreso a mi cuenta en la web de GitHub. Luego, en la interfaz, busco un botón que dice "New" o "Create repository". Ahí elijo un nombre para mi proyecto, por ejemplo "MiPrimerProyecto". También puedo agregar una descripción para recordar de qué se trata. Luego, selecciono si quiero que sea público o privado. Si quiero que tenga un archivo inicial, puedo marcar la opción "Initialize this repository with a README". Finalmente, le doy clic en "Create" y ya tendria mi nuevo repositorio listo para usar. Es como crear una carpeta en línea donde voy a guardar y gestionar mi código.

¿Cómo crear una rama en Git?

Para mí, una rama es como una línea paralela de trabajo dentro del mismo proyecto. Así puedo hacer modificaciones sin que afecten siempre a la versión principal. Desde la terminal, en mi repositorio local, puedo crear una rama con el comando `git branch nombre-de-la-rama`. Por ejemplo, si quiero trabajar en una función nueva, puedo crear una rama llamada "feature-x". Esto ayuda a organizar mejor mis cambios y mantener la estabilidad del código principal.

¿Cómo cambiar a una rama en Git?

Una vez creada la rama, si quiero empezar a trabajar en ella. Para eso uso `git checkout nombre-de-la-rama`. Esa instrucción me mueve a esa rama en mi entorno local, así puedo editar archivos y guardar cambios en ese contexto separado. También, si quiero crear y cambiar en un solo paso, puedo usar `git checkout -b nombre-de-la-rama`, lo cual es práctico para avanzar más rápido.

¿Cómo fusionar ramas en Git?

Supongamos que ya terminé en la rama "feature-x" y quiero que mis cambios se incorporen a la rama principal, generalmente llamada "main". Primero, tengo que estar en "main" con `git checkout main`. Luego, uso `git merge feature-x`, que combina los cambios de la rama "feature-x" en "main". Eso me permite integrar nuevas funcionalidades o correcciones una vez que estén probadas y listas. Pienso que este proceso es importante para mantener el control y organización del código.

¿Cómo crear un commit en Git?

Para guardar cambios en Git, primero selecciono qué archivos quiero incluir con `git add archivo`. Si quiero agregar todo, uso `git add .`. Después, hago el "guardado" de esos cambios con `git commit -m "Descripción del cambio"`. Es como decirle a Git: "Estos cambios ya están listos y revisados." Esto ayuda a mantener un historial completo de la evolución del proyecto, y puedo volver a versiones anteriores si fuera necesario.

¿Cómo enviar un commit a GitHub?

Luego de hacer commit en mi equipo, si quiero subir esos cambios a la nube (GitHub), utilizo `git push origin nombre-de-la-rama`. Ese comando manda los cambios al repositorio remoto, permitiendo que otros puedan ver y trabajar con la última versión. . Es fundamental para colaborar en equipo y tener respaldo de nuestro trabajo.

¿Qué es un repositorio remoto?

Un repositorio remoto es simplemente la copia en línea de nuestro proyecto, almacenada en plataformas como GitHub. Es como una copia en la nube donde podemos subir nuestras versiones, y desde donde otros pueden acceder a nuestro código si así lo permitimos. Nos ayuda a trabajar con varias personas, sincronizar cambios y mantener el proyecto seguro y accesible desde cualquier lugar.

¿Cómo agregar un repositorio remoto a Git?

Para conectar mi repositorio local con uno en línea, uso el comando `git remote add origin URL`. Donde "URL" es la dirección del repositorio en GitHub. Esto le dice a Git que "origin" es mi repositorio en línea, así en adelante puedo subir o bajar cambios con `git push` y `git pull` de forma sencilla.

¿Cómo empujar cambios a un repositorio remoto?

Ya que tengo los cambios en mi equipo, para enviarlos a GitHub uso `git push origin nombre-de-la-rama`. Esto asegura que mi copia en línea tenga la última versión del código que acabo de trabajar, facilitando la colaboración y respaldo.

¿Cómo tirar de cambios de un repositorio remoto?

Por otro lado, si otros hicieron cambios en el repositorio en línea, puedo actualizarlos en mi máquina con `git pull origin nombre-de-la-rama`. Es similar sincronizar mi

copia con la versión más actualizada, para no perder los avances de los demás y trabajar siempre con la versión más reciente.

¿Qué es un fork de repositorio?

Un fork es muy parecido a como hacer una copia de un proyecto de otra persona en mi propia cuenta. También me sirve para probar cosas sin tocar el original, o para contribuir en proyectos externos. Es como clonar ese proyecto, pero en mi espacio personal, donde puedo hacer cambios, mejorar o corregir y, si quiero, enviar esos cambios de vuelta al original mediante un pull request.

¿Cómo crear un fork de un repositorio?

Desde la interfaz web de GitHub, voy a la página del repositorio que me interesa, por ejemplo de un proyecto abierto. Allí, simplemente hago clic en el botón "Fork" en la esquina superior derecha. GitHub creará una copia exacta en mi cuenta, que puedo clonar y modificar libremente. Esto me permite contribuir en proyectos de otros sin afectar el código original.

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

Para enviar una pull request, primero realizo cambios en una rama diferente a la principal, generalmente en una rama de características. Cuando termino, desde la interfaz de GitHub clickeo en "New pull request", selecciono la rama donde hice los cambios y comparo con la rama principal. Luego, hago clic en "Create pull request", añado una descripción explicando qué cambios hice y por qué, y finalmente envío la solicitud para que los responsable revisen y puedan fusionar mis cambios si están de acuerdo. Esto facilita que otros revisen mi trabajo antes de integrarlo al proyecto principal.

¿Cómo aceptar una solicitud de extracción?

Como colaborador o responsable del repositorio, reviso las solicitudes en la pestaña de pull requests. Cuando veo una que me parece adecuada, reviso los cambios y, si no hay problemas, hago clic en "Merge pull request" y después en "Confirm merge" para incorporar los cambios en la rama principal. Así, garantizo que las modificaciones que acepto pasan por revisión y están validadas.

¿Qué es una etiqueta en Git?

Una etiqueta en Git es similar a un marcador que se pone en un commit específico para señalar versiones importantes, como lanzamientos de productos. Esto ayuda a tener un punto de referencia en el historial, especialmente cuando libero versiones de software y quiero volver rápidamente a esa versión concreta en el futuro.

¿Cómo crear una etiqueta en Git?

Creo una etiqueta mediante el comando `git tag v1.0` en la terminal, estando en el commit donde quiero marcar esa versión. Si quiero marcar un commit en particular,

utilizo su hash. Luego, esa etiqueta aparece en mi historia y puedo subirla a GitHub para que otros la vean.

¿Cómo enviar una etiqueta a GitHub?

Para que la etiqueta creada en mi repositorio local sea visible en GitHub, uso `git push origin v1.0`. Esto sube esa etiqueta específica al repositorio remoto en la nube, facilitando que otros la conozcan y utilicen.

¿Qué es un historial de Git?

El historial de Git es como una línea de tiempo que guarda todos los cambios, commits y fusiones que hice desde que creé el repositorio. Es útil para revisar cómo evolucionó el proyecto, qué cambios hice en qué momento, y quién los realizó.

¿Cómo ver el historial de Git?

Utilizo el comando `git log` en la terminal, que me muestra todos los commits en orden cronológico inverso, con detalles como quién hizo cada cambio, cuándo, y qué mensaje dejó. Es muy útil para entender qué cambios se han hecho y cuándo.

¿Cómo buscar en el historial de Git?

Para encontrar cambios específicos, uso `git log --grep="palabra"`. Esto filtra los commits cuyo mensaje contiene esa palabra, ayudándome a localizar cambios relacionados con un tema en particular.

¿Cómo borrar el historial de Git?

Para borrar el historial de Git, aunque sé que no es recomendable borrar el historial, pero si quisiera hacerlo, tendría que crear un nuevo repositorio desde cero, por ejemplo, haciendo un nuevo branch sin historial y luego forzar su publicación en GitHub. Sin embargo, esto puede eliminar toda la información previa, por lo que solo lo haría en casos muy especiales y sabiendo que perdería todo el historial anterior.

¿Qué es un repositorio privado en GitHub?

Un repositorio privado es aquel en el que solo las personas a las que yo les doy permisos pueden ver o modificar su contenido. Esto es útil cuando trabajo en proyectos personales o en algo sensible que no quiero que otros vean públicamente.

¿Cómo crear un repositorio privado en GitHub?

Al crear un nuevo repositorio, selecciono la opción "Private". Luego ingreso los datos del proyecto y clickeo en "Create repository". De esta forma, solo quienes invito podrán acceder al código.

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

Desde la configuración del repositorio, en la sección "Manage access", hago clic en "Invite a collaborator", ingreso su usuario de GitHub, y envío la invitación. Cuando acepte, podrá acceder y colaborar según los permisos que le otorgue.

¿Qué es un repositorio público en GitHub?

Es un repositorio accesible para todo el mundo. Cualquier usuario puede verlo, copiarlo o descargarlo simplemente visitando la URL. Es muy útil para proyectos de código abierto donde quiero compartir mi trabajo ampliamente.

¿Cómo crear un repositorio público en GitHub?

Para crear un repositorio público en GitHub realizo los siguientes pasos:

Ingreso a la cuenta en GitHub.

1. Hago clic en el botón **"New"** o **"Create repository"** (generalmente en la esquina superior derecha, o en la página principal si es tu primera creación).
2. En la sección de creación del repositorio, completo los datos:

- **Repository name:** Asigno un nombre a tu repositorio.
- **Description (opcional):** Agrego una descripción del proyecto.

4. En la opción **"Visibility"**, selecciono **"Public"** para que cualquiera pueda acceder a tu código sin restricciones.
5. Se Marca la opción **"Initialize this repository with a README"** si queres que tenga un archivo README por defecto.
6. Hago clic en **"Create repository"**.

De esta manera, el repositorio será accesible públicamente para cualquier persona en GitHub, permitiendo que puedan ver, clonar o hacer contribuciones según los permisos que configure.

¿Cómo compartir un repositorio público en GitHub?

Para poder hacerlo solo comparto la URL del repositorio, por ejemplo `https://github.com/usuario/proyecto`, y así otros pueden verlo, clonarlo o colaborar si les doy permisos. Es una manera sencilla y efectiva de difundir mi trabajo.