UNIVERSIDAD TECNOLÓGICA NACIONAL
Facultad Regional Avellaneda
Tecnicatura Universitaria en Programación
Modalidad a Distancia – Ciclo Lectivo 2025

PRIMER REPOSITORIO GIT

Asignatura: Programación I

Unidad: Unidad 2 – Trabajo Práctico

Nombres: Agustina Milagros

Apellido: Cruz

Matricula: 100652

1. ¿Qué es GitHub?

1. Es una plataforma que ofrece alojamiento de repositorios de control de versiones, que permite a los

desarrolladores almacenar y gestionar sus proyectos de software. Es una herramienta gratuita y de código

abierto que permite el trabajo en equipo en proyectos, el intercambio de código y el trabajo conjunto de

forma eficiente.

2. ¿Cómo crear un repositorio en GitHub?

2. Para crear un repositorio en GitHub, primero debo crear una cuenta en la plataforma y configurar mi perfil. Una vez que tengo mi cuenta, puedo empezar a crear repositorios. Si es la primera vez que utilizo GitHub, debo hacer clic en el botón New para crear un nuevo repositorio. Luego, completo el nombre del repositorio, una descripción breve de lo que voy a subir, y elijo si quiero que sea público o privado. Cuando todo esté listo, hago clic en Create Repository.

Después de crear el repositorio, GitHub me proporcionará diferentes comandos según mi situación. Si estoy comenzando desde cero, GitHub me dará las instrucciones para vincular mi repositorio local con el repositorio en GitHub. En este caso, debo usar los comandos que me indican para enlazar ambos repositorios. Si ya tengo un repositorio local y quiero enviarlo a GitHub, debo

seguir los comandos para agregar el repositorio remoto y luego subir los cambios con:

$ git remote add origin https://github.com/agustinamilagroscruz/PrimerRepositorio.git

$ git push -u origin master

Una vez que ejecute estos comandos, solo tengo que refrescar la página en GitHub para ver que el repositorio se ha subido correctamente.

Si en algún momento necesito trabajar en otro equipo, puedo clonar el repositorio a esa computadora. Solo debo hacer clic en Clone or download en GitHub, copiar la URL y ejecutar el siguiente comando en la terminal, en la carpeta donde quiero clonar el proyecto:

$ git clone https://github.com/agustinamilagroscruz/PrimerRepositorio.git

Después de clonar el repositorio, puedo hacer cambios en los archivos, agregarlos al stage, hacer un commit y luego subir los cambios a GitHub con el siguiente comando:

$ git push origin master

De esta manera, los cambios realizados en mi repositorio local se reflejarán en el repositorio de GitHub. Es importante recordar que, si estoy trabajando en una computadora diferente, necesitaré ingresar mi usuario y contraseña de GitHub para poder subir los cambios.

3. ¿Cómo crear una rama en Git?

3. La rama "master" en Git no es una rama especial, simplemente es la rama predeterminada que se crea cuando inicializo un repositorio con el comando git init. La razón por la que "master" aparece en casi todos los repositorios es porque es la que se genera por defecto.

Cuando quiero crear una nueva rama, utilizo el comando git branch para crearla. Sin embargo, al ejecutar este comando, no me muevo automáticamente a la nueva rama, sino que sigo estando en la rama "master". Git sabe en qué rama estoy gracias al apuntador especial denominado HEAD. Es importante recordar que aunque haya creado una nueva rama, seguiré trabajando en "master" hasta que explícitamente cambie a la nueva rama utilizando el comando git checkout o, en versiones más recientes de Git, git switch. De esta manera, me moveré de la rama "master" a la nueva rama que acabo de crear. Una vez que esté en la nueva rama, cualquier cambio que realice se registrará en esa rama y no en "master".

4. ¿Cómo cambiar a una rama en Git?

4. Para saltar de una rama a otra, utilizo el comando git checkout seguido del nombre de la rama a la que quiero cambiar. Por ejemplo:

git checkout nuevaRama

Este comando mueve el apuntador HEAD a la rama "nuevaRama", cambiándome de rama. Si lo que quiero es crear una nueva rama y cambiarme a ella en un solo paso, utilizo git checkout con la opción -b. Así, de esta manera puedo crear la nueva rama y automáticamente cambiarme a ella:

git checkout -b nuevaRama

De esta forma, ya estoy trabajando directamente en la nueva rama sin tener que ejecutar varios comandos.

5. ¿Cómo fusionar ramas en Git?

5. Para fusionar ramas en Git, lo que hago es un proceso que combina los cambios de una rama en otra. Primero, debo asegurarme de estar en la rama a la que quiero fusionar los cambios. Generalmente, esta es la rama principal, que suele ser "master" o "main", o cualquier otra rama en la que esté integrando los cambios.

Por ejemplo, si quiero fusionar la rama ‘ramaNueva’ con la rama principal “master”, primero me cambio a la rama principal:

git checkout master

Una vez que estoy en la rama de destino, utilizo el comando git merge para fusionar la rama de origen (en este caso, "nuevaRama") en la rama actual (master):

git merge nuevaRama

Este comando incorpora todos los cambios de “nuevaRama” en “master”. De esta manera, los cambios de la rama “nuevaRama” quedan integrados en la rama principal.

6. ¿Cómo crear un commit en Git?

6. Cuando quiero guardar los cambios realizados en mi repositorio, hago un commit en Git. Un commit captura el estado actual de mi código en ese momento y lo almacena en el historial del proyecto.

Primero, realizo las modificaciones en los archivos de mi proyecto. Luego, para preparar esos cambios antes de guardarlos, los agrego al área de preparación con el comando git add. Si quiero agregar un archivo específico, utilizo:

git add archivo1

Si prefiero agregar todos los archivos modificados de una vez, uso:

git add .

Una vez que los cambios están en el área de preparación, realizo el commit con el comando git commit, acompañado de un mensaje que describa los cambios hechos:

git commit -m "Mensaje de los cambios"

De esta manera, el estado actual de mi código queda guardado en el historial del repositorio con el mensaje correspondiente, lo que me permite llevar un control claro de las modificaciones realizadas.

7. ¿Cómo enviar un commit a GitHub?

7. Para enviar un commit a GitHub, primero clono mi repositorio utilizando el enlace correspondiente. En mi caso, sería:

git clone https://github.com/agustinamilagroscruz/PrimerRepositorio.git

Luego, ingreso a la carpeta del repositorio con cd PrimerRepositorio y realizo los cambios necesarios en los archivos del proyecto.

Una vez que tengo los cambios listos, los agrego al área de preparación con git add nombre_del_archivo si quiero agregar un archivo específico o git add . para agregar todos los archivos modificados.

Después, creo un commit para guardar esos cambios en mi historial local usando:

git commit -m "Descripción de los cambios"

Este mensaje debe ser claro y describir qué modificaciones realicé.

Finalmente, para que los cambios también se reflejen en GitHub, los subo al repositorio remoto con:

git push origin main

(si mi rama principal es "main") o

git push origin master

(si mi rama principal es "master").

De esta manera, mis cambios quedan registrados y disponibles en GitHub, permitiéndome acceder a ellos desde cualquier lugar o compartirlos con otros colaboradores.

8. ¿Qué es un repositorio remoto?

8. Para poder colaborar en cualquier proyecto Git, se necesita saber cómo gestionar repositorios remotos. Los repositorios remotos son versiones de tu proyecto que están hospedadas en Internet o en cualquier otra red. Puedes tener varios de ellos, y en cada uno tendrás generalmente permisos de solo lectura o de lectura

y escritura. Colaborar con otras personas implica gestionar estos repositorios remotos enviando y trayendo datos de ellos cada vez que necesites compartir tu trabajo. Gestionar repositorios remotos incluye saber cómo añadir un repositorio remoto, eliminar los remotos que ya no son válidos, gestionar varias ramas remotas, definir si deben rastrearse o no y más.

9. ¿Cómo agregar un repositorio remoto a Git?

9. Para vincular mi repositorio local con un repositorio remoto, utilizo el comando git remote add, donde le asigno un nombre al remoto para referenciarlo más fácilmente en lugar de usar la URL completa. En mi caso, si quiero vincular mi repositorio local con GitHub, escribiría:

git remote add origin https://github.com/agustinamilagroscruz/PrimerRepositorio.git

Asignarle el nombre "origin" me permite referenciarlo en otros comandos sin necesidad de escribir la URL completa.

Para asegurarme de que el remoto se agregó correctamente y verificar su nombre, ejecuto:

git remote -v

Esto me muestra una lista con los remotos configurados y sus respectivas URLs.

Si necesito traer la información del repositorio remoto sin fusionarla aún con mi rama actual, utilizo el comando git fetch seguido del nombre del remoto. En mi caso:

git fetch origin

Este comando descarga los cambios más recientes del remoto, pero no los combina automáticamente con mi código local. Es útil cuando quiero ver qué modificaciones hay antes de integrarlas a mi proyecto.

10. ¿Cómo empujar cambios a un repositorio remoto?

10. Antes de empujar mis cambios, es una buena práctica obtener los últimos cambios del repositorio remoto para evitar conflictos. Para ello, ejecuto el comando:

git pull origin nuevaRama

Este comando me asegura que mi repositorio local esté actualizado con los cambios más recientes del remoto.

Una vez que tengo todo actualizado y no tengo conflictos, empujo mis cambios al repositorio remoto con el siguiente comando:

git push origin nuevaRama

11. ¿Cómo tirar de cambios de un repositorio remoto?

11. Para tirar los cambios de un repositorio remoto, uso el comando git pull. Este comando descarga y fusiona los cambios del repositorio remoto con mi rama local.

Si estoy trabajando en la rama principal (por ejemplo, main), debo ejecutar:

git pull origin main

Este comando asegura que mi repositorio local esté actualizado con los cambios más recientes del repositorio remoto en la rama main. Si estoy trabajando en otra rama, simplemente cambio main por el nombre de la rama correspondiente.

12. ¿Qué es un fork de repositorio?

12. Un fork de repositorio es una copia de un repositorio que me permite hacer modificaciones sin afectar el repositorio original. Esto es útil cuando quiero

trabajar en un proyecto de código abierto o en algo que me interesa, pero no quiero modificar directamente el proyecto original.

Para hacer un fork, busco un repositorio que me guste en GitHub. En la página del repositorio, encuentro el botón "Fork" en la parte superior derecha. Al hacer clic en este botón, GitHub crea una copia del repositorio en mi propia cuenta. Esta copia es independiente, por lo que cualquier cambio que haga en ella no afectará al repositorio original. Luego, simplemente clono mi fork a mi máquina local para seguir trabajando en el proyecto desde allí.

De esta manera, puedo modificar el repositorio como quiera sin interferir con el trabajo de los demás.

13. ¿Cómo crear un fork de un repositorio?

13. Para crear un fork de un repositorio, puedo comenzar creando una cuenta adicional en GitHub si quiero separar mis pruebas de mi cuenta principal. Luego, busco un repositorio en mi cuenta original que quiero modificar y desde mi cuenta adicional, hago un fork de ese repositorio. Esto creará una copia en mi cuenta adicional.

Una vez que tengo el fork, lo clono en mi máquina local y me meto en el repositorio clonado. Realizo los cambios que creo necesarios en los archivos. Después, agrego esos cambios al área de preparación con git add, hago el commit con git commit y finalmente subo esos cambios a mi fork en GitHub con el comando git push origin master.

Sin embargo, esos cambios no se verán reflejados en el repositorio original, ya que estoy trabajando en una copia. Si quiero que el creador del repositorio original vea mis cambios, debo enviarle una solicitud de extracción (pull request), lo que permitirá que mi cambio sea revisado y, posiblemente, fusionado al repositorio original.

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

14. Para enviar una solicitud de extracción (pull request), me dirijo a la sección de "Pull Requests" en GitHub. Allí, hago clic en "New Pull Request". En la siguiente ventana, veré un resumen de los cambios que he realizado en comparación con el repositorio original. Luego, hago clic en "Create Pull Request".

En la nueva página, tengo un espacio para poner el asunto, donde puedo incluir un mensaje breve sobre lo que estoy proponiendo. Más abajo, tengo suficiente

espacio para explicar con más detalle por qué considero que los cambios que he realizado podrían ser útiles para el repositorio original. Es aquí donde expongo las razones para que el repositorio original considere integrar mis modificaciones.

15. ¿Cómo aceptar una solicitud de extracción?

15. Para aceptar una solicitud de extracción, primero el autor del repositorio verá en su sección de "Pull requests" el mensaje que le envié. Allí, podrá revisar los cambios que propuse y si considera que son útiles, podrá fusionarlos con su repositorio. Si no hay conflictos con la rama principal (master) y está de acuerdo con los cambios, el autor puede hacer clic en el botón "Merge pull request", lo que añadirá los cambios al repositorio original.

Además, el autor tiene la posibilidad de responderme, ya sea para confirmar que aceptará los cambios, pedir aclaraciones o realizar comentarios adicionales sobre mi propuesta.

16. ¿Qué es un etiqueta en Git?

16. Una etiqueta en Git es una forma de marcar puntos específicos en el historial del repositorio como importantes. Generalmente, las utilizo para señalar versiones de lanzamiento, como "v1.0" o "v2.0". Las etiquetas me permiten identificar fácilmente estos puntos clave y tener un control sobre las versiones del proyecto. Así puedo referirme a un estado concreto del repositorio sin tener que recordar el hash completo de ese commit.

17. ¿Cómo crear una etiqueta en Git?

17. Para crear una etiqueta en Git, puedo optar por dos tipos principales: etiquetas ligeras y etiquetas anotadas. Las etiquetas ligeras son simplemente punteros a un commit específico, como una rama que no cambia, y no contienen mucha información adicional. Por otro lado, las etiquetas anotadas se guardan en la base de datos de Git como objetos completos, incluyendo información como el nombre del etiquetador, correo electrónico, fecha, un mensaje asociado y hasta pueden ser firmadas y verificadas con GPG.

Por lo general, se recomienda crear etiquetas anotadas porque contienen más información y son más completas. Sin embargo, si necesito una etiqueta temporal o no me interesa toda esa información adicional, puedo optar por una etiqueta ligera. Para crear una etiqueta anotada, usaría el siguiente comando:

git tag -a v1.0 -m "Versión 1.0"

Y para una etiqueta ligera: git tag v1.0

De esta forma, puedo gestionar las versiones de mi proyecto de manera eficiente.Para crear una etiqueta en Git, puedo usar dos tipos principales: etiquetas ligeras y etiquetas anotadas.

Las etiquetas ligeras son similares a una rama, pero no cambian; simplemente son punteros a un commit específico. Por otro lado, las etiquetas anotadas se guardan en la base de datos de Git como objetos completos. Estas etiquetas contienen un checksum, el nombre del etiquetador, su correo electrónico, la fecha, un mensaje asociado, y pueden ser firmadas y verificadas con GNU Privacy Guard (GPG).

Generalmente, se recomienda crear etiquetas anotadas, ya que contienen más información. Sin embargo, si necesito una etiqueta temporal o por alguna razón no me interesa esa información, puedo usar etiquetas ligeras.

18. ¿Cómo enviar una etiqueta a GitHub?

18. Para enviar una etiqueta a GitHub, primero debo crear una etiqueta en mi repositorio local. Puedo crear una etiqueta anotada, que incluye información adicional como el autor y la fecha, o una etiqueta ligera, que simplemente es un puntero a un commit. Por ejemplo, para crear una etiqueta ligera, usaría el siguiente comando:

git tag v1.0

Una vez que he creado la etiqueta en mi repositorio local, necesito empujarla al repositorio remoto en GitHub. Para hacer esto, utilizo el siguiente comando:

git push origin v1.0

Aquí, origin es el nombre del repositorio remoto (que por defecto suele ser origin) y

v1.0 es el nombre de la etiqueta.

Si quiero empujar todas las etiquetas creadas de una vez, utilizo:

git push origin --tags

Además, puedo verificar las etiquetas que he creado localmente con:

git tag

19. ¿Qué es un historial de Git?

19. El historial de Git es una secuencia de todos los cambios realizados en un repositorio de Git. Cada cambio en el repositorio se guarda como un commit, y cada commit contiene información sobre el estado del proyecto en un momento específico, incluyendo:

-Identificador del commit.

-Autor.

-Fecha de realización.

-Mensaje enviado.

20. ¿Cómo ver el historial de Git?

20. Para ver el historial de Git, utilizo el comando git log. Al ejecutar este comando en el bash de Git, puedo ver el historial de commits, estando ubicado en la carpeta de mi proyecto. Los commits aparecerán en orden invertido, es decir, los más recientes se muestran primero. También puedo usar git log --oneline para una visualización más compacta, que muestra un resumen conciso de los commits recientes, con cada commit en una sola línea. Si mi proyecto tiene muchos commits, puedo optar por ver solo una cantidad específica de ellos utilizando la opción - seguida del número, como git log -3 para los últimos tres commits. Además, si quiero ver los cambios realizados en el código de cada commit, puedo usar la opción -p, lo que generará una salida más extensa que puedo navegar con los cursores, y para salir de la visualización, puedo usar CTRL + Z.

21. ¿Cómo buscar en el historial de Git?

21. Para buscar en el historial de Git, puedo utilizar varios comandos y opciones que me permiten filtrar y localizar commits específicos. Si quiero buscar commits que contengan una palabra o frase en el mensaje de commit, uso git log --grep="palabra clave". Si necesito buscar commits que han modificado un archivo en particular, puedo usar git log -- nombre_del_archivo. También puedo buscar commits dentro de un rango de fechas específico con las opciones --since y --until, como git log --since="2024-01-01" --until="2024-01-31". Y si quiero encontrar commits hechos por un autor específico, utilizo git log --author="Nombre del Autor".

22. ¿Cómo borrar el historial de Git?

22. Para borrar el historial de Git, utilizo el comando git reset, que se utiliza para deshacer cambios. Este comando mueve el puntero HEAD y, dependiendo de las opciones que elija, puede afectar el índice (área de ensayo) y el directorio de trabajo. Si utilizo la opción --hard, el comando también puede borrar cambios en el directorio de trabajo, lo que puede causar la pérdida de datos, por lo que debo tener cuidado al usarlo. Hay varias formas de usar git reset. Por ejemplo, git reset quita todos los archivos del stage, git reset nombreArchivo elimina un archivo específico del stage, git reset nombreCarpeta/ elimina todos los archivos de una carpeta del stage, git reset nombreCarpeta/nombreArchivo elimina un archivo específico dentro de una carpeta, y git reset nombreCarpeta/*.extensión elimina todos los archivos con una extensión específica dentro de una carpeta.

23. ¿Qué es un repositorio privado en GitHub?

23. Un repositorio privado en GitHub es un tipo de repositorio en el que el contenido solo es accesible para usuarios específicos que han sido autorizados. A diferencia de los repositorios públicos, donde cualquier persona puede ver y clonar el contenido, un repositorio privado limita el acceso a los colaboradores que tú elijas. Esto es útil para proyectos que contienen información sensible o que aún están en desarrollo y no deseas que estén disponibles públicamente.

24. ¿Cómo crear un repositorio privado en GitHub?

24. Para crear un repositorio privado en GitHub, primero inicio sesión en mi cuenta de GitHub. Luego, voy a la página de creación de repositorios haciendo clic en el botón "+" en la esquina superior derecha de la página principal y selecciono "New Repository" o simplemente hago clic en "New". En la siguiente pantalla, completo la información del repositorio, como el nombre y la descripción. Finalmente, selecciono la opción de privacidad "Private" para asegurar que el repositorio sea privado y solo accesible para los colaboradores que elija.

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

25. Para invitar a alguien a un repositorio privado en GitHub, primero accedo al repositorio y hago clic en la pestaña "Settings", ubicada en la parte superior del repositorio, junto a las opciones como "Code" e "Issues". Luego, en el menú de la izquierda, selecciono "Collaborators", lo cual me lleva a la página de administración de colaboradores. Allí, en la sección "Collaborators", hago clic en "Add people", luego ingreso el nombre de usuario de GitHub de la persona que quiero invitar. Después, selecciono el nivel de acceso que quiero otorgar, como

Read, Triage, Write, Maintain o Admin, y finalmente hago clic en el botón "Add" para enviar la invitación.

26. ¿Qué es un repositorio público en GitHub?

26. Un repositorio público en GitHub es un repositorio cuyo contenido es accesible a cualquier persona en Internet. A diferencia de un repositorio privado, que está restringido a un grupo específico de colaboradores, un repositorio público permite que cualquier persona pueda ver, clonar y, si tienen los permisos adecuados, contribuir al proyecto.

27. ¿Cómo crear un repositorio público en GitHub?

27. Para crear un repositorio público en GitHub, lo primero que hago es iniciar sesión en mi cuenta de GitHub. Luego, en la esquina superior derecha de la página principal, hago clic en el botón "+" y selecciono "New Repository". Esto me lleva a la página de creación de un nuevo repositorio. Allí, completo la información básica del repositorio, como el nombre y la descripción. En la sección de "Privacy", selecciono la opción "Public" para que el repositorio sea accesible para cualquier persona. Después, puedo decidir si quiero inicializar el repositorio con un README o agregar un archivo .gitignore y una licencia. Finalmente, hago clic en el botón "Create Repository" para finalizar el proceso y crear el repositorio público.

28. ¿Cómo compartir un repositorio público en GitHub?

28. La forma más sencilla de compartir tu repositorio es proporcionando el enlace directo al mismo. Para hacerlo, accedo a mi repositorio en GitHub y copio la URL que se encuentra en un cuadro de texto donde dice "Code". Puedo copiar la URL directamente haciendo clic en el botón de copiar a la derecha de la URL. Luego, ya con la URL copiada, puedo compartirla con otras personas para que accedan a mi repositorio.