Práctico 2: Git y GitHub

Objetivo:

El estudiante desarrollará competencias para trabajar con Git y GitHub, aplicando conceptos fundamentales de control de versiones, colaboración en proyectos y resolución de conflictos, en un entorno simulado y guiado.

Romero Alani Elvio Nahuel.

Resultados de aprendizaje:

1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.
2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto.
3. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo.
4. Resolver conflictos en un entorno de control de versiones: Identificar, analizar y solucionar conflictos de merge generados en un flujo de trabajo con múltiples ramas.

Actividades

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

- **¿Qué es GitHub?**

GitHub es una plataforma en línea donde podemos compartir y gestionar nuestros repositorios de forma pública o privada. Se utiliza principalmente para almacenar y trabajar en proyectos de programación. Además de alojar nuestros proyectos, GitHub permite a los usuarios ver, clonar y guardar repositorios de otros usuarios. También es una comunidad de desarrolladores que facilita la colaboración y el control de versiones, basado en Git, un sistema que permite a los desarrolladores llevar un registro de los cambios en su código y trabajar de manera eficiente en proyectos de software. Esto facilita que múltiples personas puedan colaborar en el mismo proyecto de manera organizada.

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

Para crear un repositorio en GitHub, primero debemos acceder a la página oficial de GitHub e iniciar sesión. Una vez dentro, nos dirigimos a la parte superior derecha, donde veremos un icono de "+". Al hacer clic en él, se abrirá un menú con la opción de "New repository" (Nuevo repositorio). Al seleccionar esta opción, se abrirá una página con varias configuraciones.

a) Nombre del repositorio: Debemos ingresar un nombre único para nuestro repositorio.
b) Descripción (opcional): Podemos añadir una breve descripción de nuestro proyecto, aunque esto es opcional.

c) Visibilidad: Elegimos si el repositorio será público (accesible por cualquiera) o privado (solo accesible por nosotros o las personas que invitemos).
d) Opciones adicionales: Podemos decidir si queremos inicializar el repositorio con un README (archivo que describe el proyecto), un **.** gitignore (para excluir ciertos archivos) o una licencia.

Finalmente, hacemos clic en el botón "Create repository" (Crear repositorio) para completar el proceso.

- ## ¿Cómo crear una rama en Git?

  Para crear una rama en Git, utilizamos el siguiente comando:

  git branch nombre-de-la-rama

  Este comando creará una nueva rama, pero no te cambiará automáticamente a ella**.** Si deseas crear una rama y cambiarte automáticamente a ella, debes usar el siguiente comando:

  git checkout -b nombre-de-la-rama

- ## ¿Cómo cambiar a una rama en Git?

  Para cambiar a una rama en Git, si ya has creado la rama con el comando git branch nombre-de-la-rama y deseas cambiarte a esa rama, utilizas el siguiente comando:

  git checkout nombre-de-la-rama

  Este comando te cambiará a la rama especificada. Simplemente reemplaza “nombre-de-la-rama” por el nombre de la rama a la que quieras cambiarte.

- ## ¿Cómo fusionar ramas en Git?

  Para fusionar ramas en Git, primero asegúrate de estar en la rama a la que deseas fusionar los cambios (por lo general, esta es la rama main o master, pero puede ser cualquier otra rama en la que estés trabajando). Luego, usa el siguiente comando:

  Git merge “nombre-de-la-rama”

  Este comando fusionará los cambios de la rama **nombre-de-la-rama** en la rama en la que te encuentras actualmente.

- ## ¿Cómo crear un commit en Git?

  Antes de crear el commit, primero debemos preparar los archivos, necesitamos agregarlos al escenario de Git, que es el lugar donde queremos que estos sean sometidos para que Git registre esos cambios. Para realizar este paso, usamos el comando:
  Git add .
  Una vez realizado ya estamos preparados para realizar nuestro primer commit.

Para crear un commit en Git, utilizamos el siguiente comando:

Git commit -m “mensaje”

Este comando crea un commit con un mensaje que describe los cambios realizados. El mensaje entre comillas debe ser una breve descripción de lo que se ha hecho en el commit, para que otros (y tú mismo) puedan entender fácilmente qué cambios se hicieron.

- ## ¿Cómo enviar un commit a GitHub?

  Para enviar un commit a nuestro repositorio de GitHub, utilizamos el comando git push -u origin main

  (o git push -u origin master dependiendo de la rama principal de tu repositorio). Este comando envía los cambios del repositorio local al repositorio remoto en GitHub. La opción -u establece una relación de seguimiento entre la rama local y la remota, de modo que en futuros git push ya no sea necesario especificar la rama. Si es la primera vez que empujas a esa rama, utiliza git push -u origin main (o git push -u origin master). En posteriores ocasiones, basta con git push

- ## ¿Qué es un repositorio remoto?

  Un repositorio remoto es una versión de tu proyecto almacenada en internet o en una red, que permite compartir y colaborar con otros desarrolladores. Es básicamente una copia de tu proyecto que está alojada en una plataforma como GitHub

- ## ¿Cómo agregar un repositorio remoto a Git?

  Una vez creado nuestro repositorio en GitHub, para agregarlo en Git debemos copiar la instrucción que nos da el repositorio recién creado. Esta instrucción la encontramos justo después de crear el repositorio:

  (git remote add origin https://github.com/RomeroNahuell/prueba.git)

  Esta misma debemos copiarla en nuestra consola.

Lo que estamos haciendo aquí es agregar la dirección de nuestro repositorio remoto, que se llamará "origin" por defecto. Una vez configurado, solo necesitaremos hacerlo una vez para vincular nuestro repositorio local con el remoto.

Lo siguiente es usar el comando: push -u origin main (o master) según corresponda

Al dar enter, si es la primera vez, nos pedirá autenticarnos (generalmente abriendo una ventana para iniciar sesión en GitHub). Después de esto, ya tendremos nuestro repositorio remoto agregado a Git.

- ## ¿Cómo empujar cambios a un repositorio remoto?

  Para agregar cambios a nuestro repositorio remoto, seguimos estos pasos:

  Primero, usamos el comando git add . para agregar todos los archivos al escenario de Git. Este paso es fundamental y siempre debe realizarse antes de hacer un commit.

  Luego, creamos un commit con el comando git commit -m "mensaje", donde agregamos una descripción breve de lo que se trabajó o modificó.

  Por último, como nuestro repositorio remoto aún no ve estos cambios, usamos el comando:

  git push -u origin main

De esta forma, los cambios se transfieren desde nuestro repositorio local al repositorio remoto.

- ## ¿Cómo tirar de cambios de un repositorio remoto?

  Para tirar de cambios de un repositorio remoto (verificar y obtener cambios realizados), debemos usar el comando:

  git pull origin main (o master)

  Este comando verifica si el repositorio fue modificado y agrega esos cambios a nuestro repositorio local

- ## ¿Qué es un fork de repositorio?

  Un fork es una copia de un repositorio creado en tu cuenta de GitHub (u otra plataforma de control de versiones), permitiendo desarrollar cambios sin afectar el repositorio original. A diferencia del "clone" que descarga el repositorio localmente en tu computadora, el fork se realiza generando una copia completa del repositorio en tu propia cuenta de GitHub.

- ## ¿Cómo crear un fork de un repositorio?

  Para crear un fork, debemos ingresar a GitHub y buscar el repositorio que queremos copiar. Una vez dentro del repositorio, hacemos clic en la opción "Fork" ubicada generalmente en la parte superior derecha de la página.

Al hacer clic en el botón Fork, se nos presentarán varias opciones, como:

- Cambiar el nombre del repositorio
- Agregar una descripción personalizada
- Seleccionar si queremos copiar todas las ramas

De esta forma, generamos una copia completa del repositorio original en nuestra cuenta de GitHub, permitiéndonos trabajar y hacer modificaciones sin afectar el proyecto original.

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

  Para enviar una solicitud de pull request primero debemos entender que es.

  Un pull request (solicitud de extracción) es una característica fundamental en la colaboración de proyectos de software que permite proponer cambios en un repositorio.

Cuando haces un pull request:

1-Estás pidiendo al propietario de un repositorio que "extraiga" (pull) tus cambios de tu rama o fork

2-Permite mostrar y discutir las modificaciones antes de integrarlas al proyecto principal

3-Es como presentar una propuesta de cambios para que otros desarrolladores la revisen.

Una vez entendido que es un pull request procedemos a ver como se hace una solicitud de extracción a un repositorio.

Primero, debes tener un fork del repositorio original en tu cuenta de GitHub.

Crea una nueva rama en tu repositorio local para hacer los cambios:

git checkout -b nombre-de-tu-nueva-rama

luego, Realiza los cambios que deseas proponer en el código.

Agrega los cambios y haz un commit:

git add .

git commit -m "Descripción de los cambios"

Ahora, Sube la rama a tu repositorio en GitHub:

git push origin nombre-de-tu-nueva-rama

por ultimo, vamos al repositorio en git hub y buscamos el botón “pull request"

Toca el botón y Se abrirá una página donde podrás:

a) Revisar los cambios que vas a proponer

b) Escribir un título descriptivo

c) Agregar un comentario explicando tus modificaciones

Haz clic en "Create pull request"

Ahora los encargados del repositorio original podrán revisar tus cambios y decidir si los incorporan al proyecto.

- ## ¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción (pull request), nos dirigimos al repositorio donde se hizo la solicitud. Buscamos la opción de pull request y, una vez ingresado, verás todos los cambios propuestos. Podrás revisar los archivos modificados, leer comentarios y discutir cambios con el solicitante de la extracción.

El propio GitHub te mostrará si existe algún conflicto de fusión. En caso de que no existan, podrás hacer la fusión de los cambios.
Para realizar la fusión, una vez hayas revisado que todo esté bien, buscamos la opción "Merge pull request", es decir, fusionar la solicitud de extracción.

Tendrás la opción de agregar un merge commit message (mensaje de fusión) de forma opcional.

Por último, le damos a "Confirm merge" para confirmar la fusión.

Luego de realizar la fusión, GitHub te dará la opción de borrar la rama del pull request. De esta forma, mantenemos un repositorio limpio y eliminamos ramas que ya no se usarán.

También existe la opción de hacerlo desde una consola, sin necesidad de entrar a GitHub, con la siguiente línea de comandos ya antes vistas:

```
git checkout main # Cambiar a la rama principal
git  pull origin rama-del-pull-request # Traer los cambios del pull request
git merge rama-del-pull-request  # Fusionar los cambios
git push origin main # Subir los cambios al repositorio remoto
```

De esta manera, hacemos todo desde la consola:

- ## ¿Qué es un etiqueta en Git?

  Una etiqueta o tag en Git es una referencia estática a un punto específico en el historial de un repositorio, utilizada para marcar versiones o puntos importantes en el desarrollo de un proyecto. Las etiquetas permanecen igual una vez creadas, es decir, no cambian ni avanzan con nuevos commits, sino que se mantienen apuntando siempre al mismo commit.

Hay dos tipos de etiquetas:

- Etiquetas anotadas: contienen información adicional como el nombre del creador, correo, fecha y un mensaje.
- Etiquetas ligeras: son simplemente un nombre asignado a un commit específico, sin información adicional.

- ## ¿Cómo crear una etiqueta en Git?

  Para crear una etiqueta en Git, primero debemos elegir el tipo de etiqueta:

  Para la etiqueta ligera : git tag v1.0

  Esto creará una etiqueta con el nombre “v1.0” .

  Para la etiqueta anotada usamos: git tag -a v1.0 -m "Primera versión estable"

  En cambio, esto creará una etiqueta llamada “v1.0” con un mensaje.

- ## ¿Cómo enviar una etiqueta a GitHub?

  Una vez creada nuestra etiqueta dentro de nuestro repositorio local, debemos empujarla al repositorio remoto(Github, en este caso), ya que al hacer un commit y subirlo, la etiqueta no se sube automáticamente con el commit.

  Para subir la etiqueta usamos el comando:

  git push origin v1.0

  De esta forma, empujamos la etiqueta “v1.0” al repositorio remoto.

  También existe la posibilidad de enviar todas las etiquetas creadas sin necesidad de hacerlos una por una, con el comando.

  git push –tags

  Este comando empuja toda etiqueta creada en el repositorio local hacia el remoto.

- ## ¿Qué es un historial de Git?

  El historial de git es un registro completo de todos los cambios realizados en un repositorio. Este historial permite ver cómo ha evolucionado el proyecto a lo largo del tiempo.

  El historial de Git permite ver los cambios realizados en un repositorio en cada archivo, comparando distintas versiones, ramas, fusiones creadas, los distintos commits realizados, entre otras cosas.

  Cada cambio en el historial es identificado por un commit único, que contiene información como el autor, la fecha, un mensaje y un identificador único (hash) del commit

- ## ¿Cómo ver el historial de Git?

  Para ver el historial usamos el comando:

  git log

  Este comando muestra una lista de todos los commits realizados en el repositorio, con detalles como el hash(identificador único) del commit, el autor, la fecha y el mensaje.

  Si lo que deseas es ver el historial de un archivo específico usamos:

  git log <nombre-del-archivo>

  Esto muestra el historial de cambios solo para ese archivo.

  Si lo que quieres es un historial mas resumida se utiliza:

  git log –oneline

  Muestra una versión más compacta del historial, con solo el hash abreviado y el mensaje del commit.

- ## ¿Cómo buscar en el historial de Git?

  Para realizar una búsqueda en el historial de Git, combinamos el comando git log con opciones de búsqueda específicas, como por ejemplo:

  Por autor: git log --author="Nombre del Autor" Esto mostrará todo los commits realizados por el autor buscado.

  Por mensaje de commit: git log --grep="arreglar bug" Con este comando, buscamos un commit que contenga una frase o palabra especifica en el mensaje del commit.

  Por fecha: git log --since="2025-03-01" --until="2025-03-25" De esta forma, encontraremos los commits realizados entre esas fechas.

  Por palabra clave: git log -S "palabra-clave" Esto buscará los commit con la palabra clave en el contenido de los archivos

  Por archivo : git log -- <ruta-del-archivo> mostrara los commits que afectaron al archivo especificado.

  De esta forma podemos realizar búsquedas dentro del historial de Git según tus necesidades.

- ## ¿Cómo borrar el historial de Git?

Para borrar el historial de Git, existen varias opciones dependiendo de lo que quieras eliminar exactamente.

1-Si lo que deseas es borrar el historial local de un repositorio, puedes usar los siguientes comando:

```
git checkout --orphan latest_branch # Crea una nueva rama sin historial
git add -A     # Agrega todos los archivos
git commit -am "Commit inicial" # Crea el primer commit de la nueva rama
git branch -D main  # Elimina la antigua rama 'main'
git branch -m main   # Renombra 'latest_branch' a 'main'
```

2-para borrar el historial local y remoto completamente

si quieres clonar el repositorio y empezar con un historial limpio, puedes usar estos comandos:

```
git clone --depth 1 URL_DEL_REPOSITORIO # Clona el repositorio con un solo commit
cd NOMBRE_DEL_REPOSITORIO  # Entra al directorio del repositorio clonado
git branch -m main  # Renombra la rama a 'main'
```

3-En caso, que se quiera borrar archivos del historial de git:

Si necesitas eliminar un archivo específico del historial de commits, puedes usar git filter-branch (aunque es recomendable usar git filter-repo si tienes muchas versiones de Git):

```
git filter-branch --force --index-filter \
 "git rm --cached --ignore-unmatch NOMBRE_DEL_ARCHIVO" \
--prune-empty --tag-name-filter cat -- --all
```

Este comando eliminará el archivo NOMBRE_DEL_ARCHIVO de todos los commits anteriores y reescribirá el historial.

4-Si quieres eliminar completamente todos los commits anteriores:

Si quieres eliminar todos los commits anteriores y empezar desde cero con un nuevo commit, puedes usar los siguientes comandos:

```
git checkout --orphan nuevo_inicio # Crea una nueva rama sin historial
git add .  # Agrega todos los archivos
git commit -m "Reinicio del repositorio"  # Realiza el primer commit
git branch -D main  # Elimina la rama 'main'
git branch -m main # Renombra la rama 'nuevo_inicio' a 'main'
git push -f origin main # Fuerza el push de la nueva rama al remoto
```

en resumen:

1- **Borrar historial local**: Usar git checkout --orphan para crear una nueva rama sin historial y luego hacer un commit inicial.

2- **Borrar historial local y remoto**: Clonar el repositorio con --depth 1 para obtener solo el historial más reciente.

3- **Eliminar archivos del historial**: Usar git filter-branch o git filter-repo para eliminar un archivo de todos los commits anteriores.

4- **Eliminar todos los commits anteriores**: Crear una nueva rama y hacer un commit inicial, eliminando la rama main y renombrando la nueva.

- ## ¿Qué es un repositorio privado en GitHub?

  Un repositorio privado de Git Hub es un espacio de almacenamiento para proyectos de software donde el acceso está restringido. Los repositorios privados solo pueden ser vistos y modificados por personas específicas que han sido invitadas o tienen permisos.

  Algunas de las características de un repositorio privado

  Control de acceso: solo los colaboradores invitados pueden ver, clonar, modificar o contribuir al código del repositorio.

  Privacidad de código: tu código fuente permanece completamente privado y no es visible para el público en general.

  Colaboración selectiva: puedes invitar específicamente a desarrolladores, compañeros de equipo o colaboradores externos que necesiten trabajar en el proyecto.

  Configuración de permisos: Git Hub permite establecer diferentes niveles de acceso (lectura, escritura, administración) para cada colaborador.

  Seguridad: es una excelente opción para proyectos sensibles o trabajos que no están listos para ser compartidos públicamente.

- ## ¿Cómo crear un repositorio privado en GitHub?

La forma para crear un repositorio privado en GitHub es sencilla:

Dentro de la página de GitHub, buscamos el ícono **"+"**. Se nos abrirá un menú de opciones y seleccionamos **"New repository"**.

Una vez adentro, veremos varias opciones, como el nombre del repositorio, descripción, entre otras. La que nos interesa es la opción para elegir si hacer el repositorio **privado** o **público**. Verás dos opciones; selecciona **"Private"**, teniendo en cuenta que, en GitHub, la versión gratuita tiene algunas limitaciones, como el número máximo de colaboradores en repositorios privados.

Una vez completado todo lo necesario, haz clic en el botón **"Create repository"**.

De esta forma, habrás creado un repositorio privado dentro de GitHub.

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

Para invitar a un colaborador a un repositorio privado, seguimos los siguientes pasos:

1- Accedemos a nuestro repositorio privado.

2- Vamos a la configuración de nuestro repositorio apretando la opción "Settings".

3- En el menú lateral izquierdo, buscamos la opción "Manage Access" o "Collaborators".

4- Dentro de este menú, apretamos la opción "Invite a Collaborator" o "Add People".

5- Dentro de esta opción, nos pedirá el nombre de usuario, dirección de correo o username.

6- Por último, enviamos la invitación, y el colaborador recibirá un correo electrónico para aceptar la invitación. Una vez que acepte, podrá acceder al repositorio privado según los permisos que se le hayan otorgado (lectura, escritura o administración).

De esta forma, invitamos a alguien a nuestro repositorio privado.

- ## ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un espacio de almacenamiento para proyectos de software que es completamente visible y accesible para cualquier persona en internet. A diferencia de los repositorios privados, los repositorios públicos pueden ser vistos, clonados y, en muchos casos, contribuidos por cualquier usuario de GitHub.

Algunas de las características de un repositorio publico

Acceso abierto: Cualquier persona puede ver el código fuente, commits, las ramas y el historial del proyecto.

Colaboración abierta: Cualquier desarrollador puede revisar el código, sugerir mejoras, reportar errores, hacer un fork, enviar pull request

Visibilidad y transparencia

Gratuito: a diferencia del privado que tiene una versión gratuita limitada, la versión publica no tiene ningún tipo de limites.

- ## ¿Cómo crear un repositorio público en GitHub?

La forma para crear un repositorio público en GitHub es sencilla:

Dentro de la página de GitHub, buscamos el ícono **"+"**. Se nos abrirá un menú de opciones y seleccionamos **"New repository"**.

Una vez adentro, veremos varias opciones, como el nombre del repositorio, descripción, entre otras. La que nos interesa es la opción para elegir si hacer el repositorio **privado** o **público**. Verás dos opciones; selecciona **"Public"**.

Una vez completado todo lo necesario, haz clic en el botón **"Create repository"**.

De esta forma, habrás creado un repositorio público dentro de GitHub.

- **¿Cómo compartir un repositorio público en GitHub?**

  Para compartir un repositorio público, simplemente copia la dirección URL del repositorio que deseas compartir. Esta URL se encuentra en la barra de direcciones de tu navegador cuando estás en el repositorio. Luego, envíala a las personas con las que deseas compartir el repositorio. Así, podrán acceder al proyecto directamente.

ACTIVIDAD 2

https://github.com/RomeroNahuell/Actividad2.git

Actividad 3

https://github.com/RomeroNahuell/conflict-exercise.git