Práctico 2

CORDERO Marina Giselle

¿Qué es GitHub?

GitHub es una plataforma que facilita la colaboración, el control de versiones y el alojamiento de código, lo que la convierte en una plataforma invaluable para proyectos de cualquier tamaño.

¿Cómo crear un repositorio en GitHub?

Desde el botón "New" (Nuevo) en la esquina superior derecha de la página, una vez allí colocar el nombre, alguna que otra descripción de lo que vamos a subir, si queremos que sea público o privado, etc, luego damos click en Create Repository

¿Cómo crear una rama en Git?

Para crear una nueva rama, se usará el comando git branch:

$ git branch nuevaRama

¿Cómo cambiar a una rama en Git?

$ git checkout nuevaRama

Para crear una nueva rama y saltar a ella, en un solo paso

$ git checkout -b nuevaRama

¿Cómo fusionar ramas en Git?

Primero, debes estar en la rama a la que quieres fusionar los cambios. Por lo general, es a la rama principal (master o main) o cualquier otra rama en la que estés integrando los cambios.

$ git merge **rama**

¿Cómo crear un commit en Git?

Antes de crear un commit, es importante verificar qué archivos han sido modificados. Para ello, utiliza el siguiente comando en la terminal:

- git status

Para agregar los cambios, utiliza el comando git add:

- Para agregar un archivo específico: git add nombre_del_archivo

- Para agregar todos los cambios en el directorio actual: git add .

Una vez que hayas agregado los cambios al área de preparación, puedes crear el commit utilizando el siguiente comando:

- git commit -m "Mensaje descriptivo del commit"

¿Cómo enviar un commit a GitHub?

Primero, verifica que tu repositorio local esté conectado al repositorio remoto en GitHub. Puedes hacerlo con el siguiente comando:

- git remote -v

- `git push origin nombre_de_la_rama`
- Reemplaza "nombre_de_la_rama" con el nombre de la rama que quieres enviar (por ejemplo, "main" o "master").

¿Qué es un repositorio remoto?

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.

¿Cómo agregar un repositorio remoto a Git?

Antes de agregar un nuevo repositorio remoto, es útil verificar si ya tienes alguno configurado. Puedes hacerlo con el siguiente comando en tu terminal:

- git remote -v

Para agregar un nuevo repositorio remoto, utiliza el siguiente comando:

- git remote add <nombre_remoto> <URL_del_repositorio>
- <nombre_remoto>: Elige un nombre para el repositorio remoto. "origin" es el nombre convencional para el repositorio remoto principal, pero puedes usar otros nombres si tienes varios repositorios remotos.

¿Cómo empujar cambios a un repositorio remoto?

Primero, es crucial asegurarse de que tu repositorio local esté conectado al repositorio remoto en GitHub (o en la plataforma que estés utilizando). Puedes verificar esto con el siguiente comando en tu terminal:

- git remote -v

Una vez que tienes el repositorio remoto configurado, puedes enviar tus commits con el siguiente comando:

- `git push origin nombre_de_la_rama`
- Reemplaza "nombre_de_la_rama" con el nombre de la rama que quieres enviar (por ejemplo, "main" o "master").

¿Cómo tirar de cambios de un repositorio remoto?

1.git fetch

- **Funcionamiento**:
  - Descarga las ramas, commits y etiquetas del repositorio remoto al repositorio local.
  - Actualiza las "ramas remotas" en tu repositorio local.
  - No fusiona automáticamente estos cambios en tus ramas locales de trabajo.
- **Uso**:
  - Para revisar los cambios remotos antes de fusionarlos.
  - Para mantener tu repositorio local actualizado con la información del remoto.
  - En entornos colaborativos, para ver los cambios de otros desarrolladores.
- **Comando**:
  - git fetch origin (descarga todos los cambios del repositorio remoto "origin").
  - git fetch origin nombre_de_la_rama (descarga solo los cambios de la rama especificada).

2. git pull

- **Funcionamiento**:
  - Descarga los cambios del repositorio remoto.
  - Fusiona automáticamente los cambios en tu rama local de trabajo.
  - Es una combinación de git fetch y git merge.
- **Uso**:
  - Para descargar y fusionar los cambios remotos directamente en tu rama local.
  - Para simplificar el proceso de actualización del repositorio local.
- **Comando**:

- git pull origin nombre_de_la_rama (descarga y fusiona los cambios de la rama especificada).

Diferencias clave

- git fetch te permite inspeccionar los cambios remotos antes de fusionarlos, mientras que git pull los fusiona automáticamente.
- git fetch es más seguro, ya que no modifica tus ramas locales de trabajo a menos que lo hagas explícitamente.
- git pull es más conveniente para actualizaciones rápidas, pero puede generar conflictos si no se usa con precaución.

Recomendaciones

- Si quieres revisar los cambios antes de integrarlos, usa git fetch seguido de git merge.
- Si quieres actualizar tu rama local rápidamente, usa git pull.
- En entornos colaborativos, es recomendable usar git fetch para evitar conflictos inesperados.

¿Qué es un fork de repositorio?

Un "fork" de un repositorio se refiere a una copia personal de un repositorio existente.

Copia completa e independiente del repositorio original, lo que significa que puedes realizar cambios sin afectar el proyecto original.

Tu fork tiene su propia URL y es un repositorio Git completo, con su propio historial de commits y ramas.

Los cambios que realices en tu fork no se reflejarán automáticamente en el repositorio original.

Puedes realizar cambios en tu fork y luego enviar una "pull request" al repositorio original para proponer tus modificaciones.

Flujo de trabajo típico:

1. **Hacer un fork:**
    - Crea una copia del repositorio original en tu cuenta de GitHub.
2. **Clonar el fork:**
    - Descarga tu fork a tu computadora local para trabajar en él.
3. **Realizar cambios:**
    - Modifica el código en tu fork local.

4. **Enviar cambios (push):**
    - Sube tus cambios a tu fork en GitHub.
5. **Crear una pull request:**
    - Solicita que el propietario del repositorio original incorpore tus cambios.

¿Cómo crear un fork de un repositorio?

1. Navega al repositorio que deseas forkar:

- Ve a la página del repositorio en GitHub. Puedes encontrarlo buscando el nombre del repositorio o accediendo a través de un enlace.

2. Haz clic en el botón "Fork":

- En la esquina superior derecha de la página del repositorio, verás un botón que dice "Fork". Haz clic en él.

3. Selecciona tu cuenta de GitHub:

- GitHub te pedirá que elijas dónde quieres crear el fork. Si tienes varias organizaciones a las que perteneces, puedes seleccionar la que prefieras. En la mayoría de los casos, seleccionarás tu cuenta personal.

4. Espera a que se cree el fork:

- GitHub comenzará a crear una copia del repositorio en tu cuenta. Este proceso puede tardar unos segundos o minutos, dependiendo del tamaño del repositorio.

5. ¡Listo!

- Una vez que se complete el proceso, serás redirigido a la página de tu fork. Verás que la URL del repositorio ha cambiado para reflejar tu nombre de usuario.

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

1. Haz un fork del repositorio:
   Si no tienes permisos de escritura en el repositorio original, necesitas hacer un fork. Ve a la página del repositorio y haz clic en el botón "Fork" en la esquina superior derecha.

2. Clona tu fork localmente:

2. En tu terminal, usa `git clone URL_de_tu_fork` para descargar tu copia del repositorio a tu computadora.

3. Crea una rama para tus cambios:

   Navega al directorio del repositorio clonado con `cd nombre_del_repositorio`.
   Crea una nueva rama para tus cambios con `git checkout -b nombre_de_tu_rama`.

4. Realiza tus cambios:
   - Edita los archivos del proyecto con tus modificaciones.

5. Confirma tus cambios (commit):
   - Agrega los archivos modificados al área de preparación con git add ..
   - Crea un commit con un mensaje descriptivo usando git commit -m "Mensaje descriptivo".

6. Envía tus cambios a tu fork en GitHub (push):
   - Sube tu rama a tu fork en GitHub con git push origin nombre_de_tu_rama.

7. Crea la solicitud de extracción (pull request):
   - Ve a tu fork en GitHub.
   - Verás un botón "Compare & pull request". Haz clic en él.
   - Revisa los cambios y asegúrate de que la rama base sea la correcta en el repositorio original.
   - Agrega un título y una descripción clara de tus cambios.
   - Haz clic en "Create pull request".

8. Espera la revisión:

¿Cómo aceptar una solicitud de extracción?

El autor del repositorio verá en sus pull requests el mensaje que le hemos enviado, para que lo pueda observar y si lo considera realizar el cambio pertinente (además de poder responderle al usuario que le ha propuesto ese cambio). Lo bueno de todo esto es que si el usuario original considera que esta modificación es buena y no genera conflictos con la rama maestra de su repositorio local remoto, puede clickear en Merge pull request y de esta manera sumará a su repositorio los cambios que hizo un usuario (en modo de ayuda).

¿Qué es un etiqueta en Git?

Una etiqueta (tag) es una forma de marcar un punto específico en la historia de tu repositorio. Es como un marcador que te permite identificar fácilmente versiones importantes de tu proyecto.

¿Cómo crear una etiqueta en Git?

Etiqueta ligera: git tag <nombre_etiqueta>

Etiqueta anotada: git tag -a <nombre_etiqueta> -m "<mensaje>"

¿Cómo enviar una etiqueta a GitHub?

Primero, debes crear una etiqueta en tu repositorio local.

Podes verificar las etiquetas que has creado localmente con: git tag

Una vez que has creado la etiqueta en tu repositorio local, necesitas empujarla al repositorio remoto en GitHub.

git push origin v1.0

¿Qué es un historial de Git?

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

¿Cómo ver el historial de Git?

git log

git log –oneline (resumen).

Si queremos que el log también nos muestre los cambios en el código de cada commit podemos usar la opción -p. Esta opción genera una salida mucho más larga, por lo que seguramente nos tocará movernos en la salida con los cursores y usaremos CTRL + Z para salir. git log -2 -p

¿Cómo buscar en el historial de Git?

Para buscar commits que contengan una palabra o frase específica en el mensaje de commit: git log --grep="palabra clave"

Para buscar commits que han modificado un archivo específico, usa git log seguido del nombre del archivo: git log -- nombre_del_archivo

Para buscar commits en un rango de fechas específico, usa las opciones --since y --until: git log --since="2024-01-01" --until="2024-01-31"

Para encontrar commits hechos por un autor específico, usa --author: git log --author="Nombre del Autor"

¿Cómo borrar el historial de Git?

Existen distintas formas de utilizarlo:

• git reset -> Quita del stage todos los archivos y carpetas del proyecto.

• git reset nombreArchivo -> Quita del stage el archivo indicado.

• git reset nombreCarpeta/ -> Quita del stage todos los archivos de esa carpeta.

• git reset nombreCarpeta/nombreArchivo -> Quita ese archivo del stage (que a la vez está dentro de una carpeta).

• git reset nombreCarpeta/*.extensión -> Quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta del stage.

¿Qué es un repositorio privado en GitHub?

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.

¿Cómo crear un repositorio privado en GitHub?

1. Inicia sesión en GitHub:

- Ve a la página de GitHub (github.com) e inicia sesión con tu cuenta.

2. Crea un nuevo repositorio:

- En la esquina superior derecha de cualquier página, haz clic en el botón "+" y selecciona "New repository".
- Se abrirá la página "Create a new repository".

3. Configura el repositorio:

- **Repository name (Nombre del repositorio):** Escribe un nombre para tu repositorio.
- **Description (Descripción):** Opcionalmente, puedes agregar una descripción para tu repositorio.
- **Public/Private (Público/Privado):** Aquí es donde eliges la visibilidad de tu repositorio. Selecciona "Private".
- **Initialize this repository with:** Puedes marcar esta casilla si deseas inicializar el repositorio con un archivo README, un archivo .gitignore o una licencia. Esto es opcional, pero recomendable.

4. Crea el repositorio:

- Haz clic en el botón "Create repository".

Información adicional:

- Los repositorios privados solo son visibles para ti y para las personas a las que les des permiso.
- GitHub ofrece diferentes planes, algunos de los cuales ofrecen más funcionalidades para repositorios privados.
- Puedes cambiar la visibilidad de un repositorio en cualquier momento, desde la configuración del mismo.

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

1. Navega a la configuración del repositorio:

- Ve a tu repositorio privado en GitHub.
- Haz clic en la pestaña "Settings" (Configuración) que se encuentra en la parte superior de la página.

2. Accede a la sección de colaboradores:

- En la barra lateral izquierda, busca y haz clic en "Collaborators & teams" (Colaboradores y equipos).

3. Invita a un colaborador:

- Haz clic en el botón verde "Add people" (Agregar personas).

- Comienza a escribir el nombre de usuario de GitHub, el nombre completo o la dirección de correo electrónico de la persona a la que deseas invitar.
- Selecciona el usuario correcto de la lista de sugerencias.
- Selecciona el rol que deseas otorgarle a esa persona. (Puede ser "Read", "Triage", "Write" o "Maintain").
- Haz clic en el botón "Add [nombre de usuario] to [nombre del repositorio]".

4. El colaborador recibe la invitación:

- GitHub enviará una notificación por correo electrónico a la persona invitada.
- La persona invitada deberá aceptar la invitación para obtener acceso al repositorio.

¿Qué es un repositorio público en GitHub?

es un espacio de almacenamiento en línea donde puedes alojar tu código, archivos y el historial de versiones de tu proyecto, con la característica principal de que **cualquier persona en internet puede verlo y acceder a él**.

¿Cómo crear un repositorio público en GitHub?

Crea un nuevo repositorio:

- En la esquina superior derecha de cualquier página, haz clic en el botón "+" y selecciona "New repository".
- Se abrirá la página "Create a new repository".

Configura el repositorio:

- **Repository name (Nombre del repositorio):** Escribe un nombre descriptivo para tu repositorio.
- **Description (Descripción):** Opcionalmente, puedes agregar una descripción breve del proyecto.
- **Public/Private (Público/Privado):** Asegúrate de que esté seleccionada la opción "Public".
- **Initialize this repository with:**

- Puedes marcar esta casilla si deseas inicializar el repositorio con:
  - Un archivo README (recomendado para describir tu proyecto).
  - Un archivo .gitignore (para ignorar archivos específicos).
  - Una licencia (para definir cómo otros pueden usar tu código).

Crea el repositorio:

- Haz clic en el botón "Create repository".

¿Cómo compartir un repositorio público en GitHub?

La forma más sencilla de compartir tu repositorio es proporcionar el enlace directo al mismo. Accede a tu repositorio, copia la URL de tu repositorio que se encuentra en un cuadro de texto que dice "<> Code”: