# Práctico 2: Git y GitHub

*Actividades*

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

- ¿Qué es GitHub?
  GitHub es una plataforma online, similar a una red social, en donde se pueden compartir nuestros proyectos, descargar, guardar y colaborar con repositorios de otras personas.

- ¿Cómo crear un repositorio en GitHub?
  En la plataforma se debe clickear en el signo “+” (esquina superior derecha de la pantalla principal) → elegí "New repository". Se le debe asignar un nombre, opcionalmente una descripción del proyecto y la visibilidad que va a tener el mismo; público para que pueda ser visto por cualquiera de la comunidad o privado para que sea visto por uno mismo o cualquiera al que se le dé acceso.

- ¿Cómo crear una rama en Git?
  Para crear una rama en Git es necesario escribir en la consola el comando *git branch “nombre_de_la_rama”*

- ¿Cómo cambiar a una rama en Git?
  Para cambiar a una rama existente en Git se utiliza el comando *git checkout “nombre_de_la_rama”*

- ¿Cómo fusionar ramas en Git?
  Para fusionar ramas en Git, primero se debe posicionar en la rama donde se desea aplicar la fusión (por ejemplo, *main*) utilizando *git checkout main*, y luego ejecutar el comando *git merge nombre_de_la_rama*.

- ¿Cómo crear un commit en Git?
  Para crear un commit en Git, primero se deben seleccionar los cambios que se desean guardar usando el comando *git add . .* Luego, se crea el commit con el comando *git commit -m "Mensaje del commit"*, donde el mensaje debe describir brevemente los cambios realizados.

- ¿Cómo enviar un commit a GitHub?
  Luego de haber creado el commit localmente con *git commit*, se utiliza el comando *git push origin nombre_de_la_rama*, donde *“origin”* es el nombre del repositorio remoto y *“nombre_de_la_rama”* es la rama en la que estás trabajando.

- ¿Qué es un repositorio remoto?
  A diferencia del repositorio local, que se encuentra en la computadora del usuario, el repositorio remoto es una versión del proyecto que está alojada en una plataforma externa, como GitHub. Este repositorio permite que varios colaboradores trabajen en el mismo proyecto desde diferentes lugares, sincronizando sus cambios.

- ¿Cómo agregar un repositorio remoto a Git?
  Para agregar un repositorio remoto a Git, se usa el comando *git remote add* seguido del nombre que se le asignará al repositorio remoto (por ejemplo, origin) y la URL del repositorio remoto.

- ¿Cómo empujar cambios a un repositorio remoto?

  Para empujar cambios a un repositorio remoto en Git, primero se debe realizar un commit con *git add* y *git commit*. Luego, se utiliza el comando *git push origin nombre_de_la_rama* para subir los cambios a GitHub. Si es la primera vez que se hace, se recomienda *git push -u origin nombre_de_la_rama* para vincular la rama local con la remota. Esto permite mantener sincronizado el trabajo local con el repositorio compartido.

- ¿Cómo tirar de cambios de un repositorio remoto?

  Para tirar (traer) cambios de un repositorio remoto, se utiliza el comando *git pull*. Este comando actualiza la rama local con los últimos cambios realizados en el repositorio remoto. Es útil para mantener actualizado tu proyecto, especialmente cuando se trabaja en equipo.

- ¿Qué es un fork de repositorio?

  Un fork es una copia de un repositorio que se crea en tu cuenta de GitHub a partir de otro repositorio existente. Permite hacer cambios libremente sin afectar el proyecto original.

- ¿Cómo crear un fork de un repositorio?

  Para crear un fork de un repositorio en GitHub, primero se debe iniciar sesión en la plataforma. Luego, se busca el repositorio que se desea copiar y se hace clic en el botón “Fork” (ubicado arriba a la derecha). GitHub crea automáticamente una copia del repositorio en tu cuenta. A partir de ahí, podés trabajar en esa copia sin afectar el proyecto original, y si querés proponer cambios, podés enviar un pull request al repositorio fuente.

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

  Para enviar una solicitud de extracción en GitHub, primero hay que hacer cambios en una rama de un fork o del mismo repositorio. Luego, se va a la página del repositorio original y aparece un botón que dice “Compare & pull request”. Al hacer clic, se abre un formulario donde se puede agregar un título y descripción de los cambios. Finalmente, se hace clic en “Create pull request” para enviar la solicitud al repositorio original, proponiendo que se integren tus cambios.

- ¿Cómo aceptar una solicitud de extracción?

  Para aceptar una solicitud de extracción en GitHub, primero se debe ingresar al repositorio que la recibió. En la pestaña "Pull requests", se selecciona la solicitud pendiente. Ahí se pueden revisar los cambios propuestos, comentar o pedir modificaciones. Si todo está bien, se hace clic en “Merge pull request”, luego en “Confirm merge”, y los cambios se integran al repositorio.

- ¿Qué es una etiqueta en Git?

  Una etiqueta en Git es una referencia que apunta a un commit específico en el historial del proyecto. Se usa principalmente para marcar puntos importantes en el desarrollo, como versiones de software. A diferencia de las ramas, las etiquetas no cambian y son inmutables. Se pueden usar para crear versiones estables o hitos importantes de un proyecto.

- ¿Cómo crear una etiqueta en Git?

  Para crear una etiqueta en Git, se utiliza el comando git tag seguido del nombre de la etiqueta. Existen etiquetas ligeras (*git tag nombre_de_etiqueta*) y las etiquetas anotadas (*git tag -a nombre_de_etiqueta -m "Mensaje descriptivo de la etiqueta*")

- ¿Cómo enviar una etiqueta a GitHub?

  Una vez que se ha creado una etiqueta en Git de forma local, es posible enviarla al repositorio remoto en GitHub para que quede registrada y accesible para otros colaboradores del proyecto. Esto es especialmente útil para marcar versiones importantes del código. *git push origin nombre_de_la_etiqueta* para enviar una única etiqueta al repositorio remoto y *git push origin --tags* en caso de querer subir todas las etiquetas creadas.

- ¿Qué es un historial de Git?

  El historial de Git es el registro completo de todos los cambios realizados en un repositorio desde su creación. Cada vez que se hace un commit, Git guarda una captura del estado del proyecto junto con información del mismo. Permite ver qué se modificó, cuándo y por quién, lo cual es fundamental para entender la evolución del código, solucionar errores o volver a versiones anteriores.

- ¿Cómo ver el historial de Git?

  *git log* es el comando que muestra una lista detallada de todos los commits realizados, incluyendo el identificador único, el autor, la fecha y el mensaje asociado a cada cambio. Es una herramienta clave para revisar el progreso del proyecto, entender cuándo se hicieron modificaciones y por quién.

- ¿Cómo buscar en el historial de Git?

  Se puede utilizar el comando *git log* junto con diferentes opciones para filtrar la información.
  - Por mensaje de commit: *git log --grep="palabra clave"*
  - Por autor: *git log --author="Nombre del autor"*
  - Por fecha: *git log --since="2024-01-01" --until="2024-12-31"*

- ¿Cómo borrar el historial de Git?

  Borrar el historial de Git significa eliminar todos los commits anteriores y comenzar con un nuevo historial desde el estado actual del proyecto.
  La mejor forma de hacerlo es reescribiendo todo el historial:

  *git checkout --orphan nueva-rama*
  *git add .*
  *git commit -m "Nuevo inicio del historial"*

  Luego se borra la rama anterior y se reemplaza:

  *git branch -D main*
  *git branch -m main*
  *git push -f origin main*

- ¿Qué es un repositorio privado en GitHub?

  Un repositorio privado en GitHub es un espacio de trabajo donde el código solo puede ser visto y accedido por las personas que el propietario autorice. El contenido de un repositorio privado está protegido y es ideal para proyectos personales, académicos o de trabajo que no deben compartirse públicamente.

- ¿Cómo crear un repositorio privado en GitHub?

  Para crear un repositorio privado es necesario hacer clic en el botón “New repository” (ubicado en la parte superior derecha en el menú desplegable con el símbolo “+”). Se completa el formulario con el nombre del repositorio, una descripción opcional y se selecciona la visibilidad. En este caso, se elige la opción “Private”, lo que garantiza que solo el creador y los colaboradores autorizados podrán acceder al contenido del repositorio. Finalmente, se hace clic en el botón “Create repository” para finalizar el proceso.

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

  Ingresá al repositorio y hay que clickear en la pestaña "Settings" en la parte superior del repositorio. Luego, en el menú lateral izquierdo, seleccionar "Manage access", ahí se encuentra la opción "Invite a collaborator". Se escribe el nombre de usuario de la persona se va a invitar y se elige el nivel de acceso que se le otorgará, ya sea solo lectura o con permisos de edición. Finalmente, clic en "Add" o "Send Invitation" para enviar la invitación.

- ¿Qué es un repositorio público en GitHub?

  Un repositorio público es un espacio de almacenamiento donde el código y todos los archivos asociados están disponibles para cualquier persona en la web. Esto significa que cualquier usuario puede ver, clonar, hacer un fork y contribuir al proyecto.

- ¿Cómo crear un repositorio público en GitHub?

  Para crear un repositorio público se repite el procedimiento para repositorios privados (clic en el botón “New repository” ubicado en la parte superior derecha en el menú desplegable con el símbolo “+”), sólo que esta vez elegimos la opción “Public” para que el repositorio sea accesible para cualquier persona.

- ¿Cómo compartir un repositorio público en GitHub?

  En la página principal del repositorio, aparece un enlace en la parte superior que muestra la URL del repositorio, que normalmente tiene el formato *https://github.com/usuario/nombre-del-repositorio*. Se puede copiar la URL y compartirla directamente con otras personas a través de cualquier medio. Al ser un repositorio público, cualquier persona con el enlace podrá acceder a él, ver el código y contribuir si el repositorio permite hacerlo.

2) Realizar la siguiente actividad:

- Crear un repositorio.
  - Dale un nombre al repositorio.
  - Elije el repositorio sea público.
  - Inicializa el repositorio con un archivo.
- Agregando un Archivo
  - Crea un archivo simple, por ejemplo, "mi-archivo.txt".
  - Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.
  - Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).
- Creando Branchs
  - Crear una Branch
  - Realizar cambios o agregar un archivo
  - Subir la Branch

*https://github.com/camilocapelli/mi-actividad-2_TP_Colaborativo.git*

3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz clic en "Create repository".

Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio (usualmente algo como https://github.com/tuusuario/conflict-exercise.git).
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando:

```
git clone https://github.com/tuusuario/conflict-exercise.git
```

- Entra en el directorio del repositorio:

```
cd conflict-exercise
```

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:

```
git checkout -b feature-branch
```

- Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo:

Este es un cambio en la feature branch.

- Guarda los cambios y haz un commit:

```
git add README.md
```

```
git commit -m "Added a line in feature-branch"
```

Paso 4: Volver a la rama principal y editar el mismo archivo

Cambia de vuelta a la rama principal (main):

```
git checkout main
```

- Edita el archivo README.md de nuevo, añadiendo una línea diferente:

Este es un cambio en la main branch.

- Guarda los cambios y haz un commit:

```
git add README.md
```

```
git commit -m "Added a line in main branch"
```

Paso 5: Hacer un merge y generar un conflicto

- Intenta hacer un merge de la feature-branch en la rama main:

```
git merge feature-branch
```

- Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md.

Paso 6: Resolver el conflicto

- Abre el archivo README.md en tu editor de texto. Verás algo similar a esto:

```
<<<<<<< HEAD
Este es un cambio en la main branch.
=======
Este es un cambio en la feature branch.
>>>>>>> feature-branch
```

- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
- Edita el archivo para resolver el conflicto y guarda los cambios(Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).
- Añade el archivo resuelto y completa el merge:

```
git add README.md
```

```
git commit -m "Resolved merge conflict"
```

Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub:

```
git push origin main
```

- También sube la feature-branch si deseas:

```
git push origin feature-branch
```

Paso 8: Verificar en GitHub

- Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.
    - Puedes revisar el historial de commits para ver el conflicto y su resolución.

*https://github.com/camilocapelli/conflict-exercise.git*