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.

*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 comunidad donde se puede compartir repositorios de forma remota publico o privado

* + - ¿Cómo crear un repositorio en GitHub?

1. Entras a GitHub y logueas en tu cuenta
2. Click en el icono “**+**” & selecciona **New repository** .
3. Completa info: Repository Name , Description , seleccionar Public
4. Click en **Create repository**
   * + ¿Cómo crear una rama en Git?

git branch <nombre\_de\_rama>

* + - ¿Cómo cambiar a una rama en Git?

git checkout <nombre\_de\_rama>

git switch <nombre\_de\_rama>

* + - ¿Cómo fusionar ramas en Git?

git merge <nombre\_de\_otra \_rama>

Si hay conflictos:

Git te va a avisar qué archivos tienen conflictos. Vas a tener que abrir esos archivos, resolver los conflictos manualmente (eliminando las marcas <<<<<<<, =======, >>>>>>>), y luego hacer:

* + - ¿Cómo crear un commit en Git?
      * Asegurate de estar en el directorio del proyecto: **cd ruta/a/tu/proyecto**
      * Agregá los archivos que querés incluir en el commit: git add .
      * Hace el commit: git commit -m "Mensaje descriptivo del cambio"
    - ¿Cómo enviar un commit a GitHub?

Una vez que hiciste un commit en Git localmente, podés enviarlo (o “pushearlo”) a GitHub para que quede guardado en el repositorio remoto.

Asegurate de estar en la rama correcta: **git branch , git checkout**

Subí tu commit al repositorio remoto: **git push origin <nombre\_de\_rama>**

* + - ¿Qué es un repositorio remoto?

Un repositorio remoto en Git es una versión de tu proyecto que está guardada en un servidor online, como GitHub, GitLab ,etc.

* + - ¿Cómo agregar un repositorio remoto a Git?

1. Primero creas un repositorio en GitHub
2. En tu proyecto local, abrí la terminal y posicionate en la carpeta: cd ruta/a/tu/proyecto
3. Agrega el repositorio remoto: git remote add origin <https://github.com/tu-usuario/mi-proyecto.git>
4. Verificá que se agregó correctamente: git remote -v
   * + ¿Cómo empujar cambios a un repositorio remoto?

git push origin nombre-de-la-rama

* + - ¿Cómo tirar de cambios de un repositorio remoto?

Tirar cambios" de un repositorio remoto en Git se llama **hacer un pull**, y sirve para **descargar los últimos cambios del servidor (como GitHub) a tu repositorio local**.

git pull origin nombre-de-la-rama

* + - ¿Qué es un fork de repositorio?

Un **fork** en GitHub es básicamente una **copia de un repositorio** que se crea **bajo tu cuenta**, pero que **mantiene un vínculo con el original**.

* + - ¿Cómo crear un fork de un repositorio?

1. Entrá al repositorio que querés forkar. Ejemplo: <https://github.com/otro-usuario/proyecto-interesante>
2. En la parte superior derecha, hacé clic en el botón “Fork” (al lado de “Star” y “Watch”).
3. GitHub te va a preguntar a qué cuenta querés forkarlo
   * + ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

Una solicitud de extracción (pull request) se usa para proponer cambios en el código de un repositorio, sobre todo cuando estás colaborando en proyectos en GitHub.

1. Tener tu **propia copia del repositorio** (ya sea un *fork* o clon).
2. Haber hecho los cambios en una **rama nueva** (recomendado).
3. Haber hecho al menos **un commit** y empujado los cambios a tu repositorio remoto.

**git push origin mi-rama**

* + - Entrá a tu repositorio en GitHub (o al original si hiciste fork).
    - GitHub va a detectar que hiciste cambios y te mostrará un botón como:
    - Compare & pull request" o "Contribute > Open pull request
    - ¿Cómo aceptar una solicitud de extracción?

Para aceptar (fusionar) una solicitud de extracción (pull request) en GitHub sigue estos pasos:

1. **Ir a la pestaña de Pull Request**: En tu repositorio de GitHub, hacé clic en la pestaña Pull requests. Verás la lista de PullRequests abiertas; hacé clic en la que quieras revisar.
2. **Revisar los cambios:** Files changed: las diferencias línea a línea, Conversation: comentarios, discusiones y aprobaciones.
3. **Fusiona el pull resquest:** Una vez aprobado, bajá hasta el botón verde Merge pull request.
4. **(OPCIONAL) eliminar rama:** despues de fusionar , GitHub te mostrara un botón para delete branch, si ya no necesitas la rama

**Create a merge commit**

* + - ¿Qué es un etiqueta en Git?

Una etiqueta es como una marca o señal que le pones a un punto importante en la historia de tu proyecto. Normalmente se usa para marcar versiones específicas del código, como por ejemplo:

* v1.0.0
* v2.3
* release-final
  + - ¿Cómo crear una etiqueta en Git?

Abre la terminal y entra a tu proyecto: cd ruta/del/proyecto

Asegurate de estar en el commit que quieres etiquetar: git log –oneline

(OPCION 1) Crea una etiqueta ligera : git tag v1.0.0 (Esto crea una etiqueta llamada v1.0.0 en el último commit.)

(OPCION 2) Crea una etiqueta con mensaje (anotada): git tag -a v1.0.0 -m "Primera versión estable"

VER TODAS LAS ETIQUETAS: git tag

Subir la etiqueta al repositorio remoto (gitHub): git push origin v1.0.0

* + - ¿Cómo enviar una etiqueta a GitHub?

**Subir la etiqueta al repositorio remoto (gitHub): git push origin v1.0.0**

* + - ¿Qué es un historial de Git?

El historial de Git es el registro de todos los cambios que se hicieron en un proyecto a lo largo del tiempo.

git log

* + - ¿Cómo ver el historial de Git?

Git log

* + - ¿Cómo buscar en el historial de Git?

1. BUSCAR POR MENSAJE DE COMMIT

git log --grep="palabra clave" ej: git log --grep="login"

1. BUSCAR CAMBIOS EN UN ARCHIVO ESPECIFICO

git log nombre-del-archivo ej: git log src/login.js

1. BUSCAR COMMITS DE UN AUTO ESPECIFICO

git log --author="nombre o email" ej : git log --author="Juan"

* + - * ¿Cómo borrar el historial de Git?
    - ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un lugar donde podés guardar tu código, pero sólo vos y las personas que vos invites pueden verlo o modificarlo.

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

Cuando crees un nuevo repositorio ponerlo en privado

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

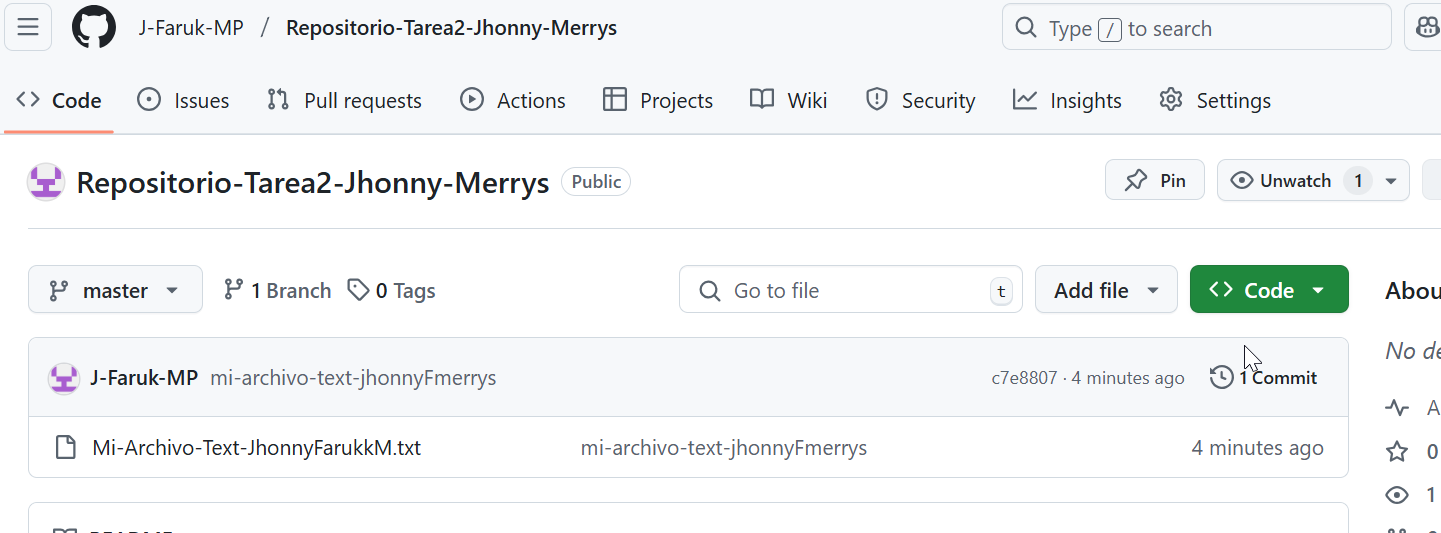
Ir a la pestaña de Setting , hace click en Collaborators 🡪 Add people🡪 Escribe nombre de usuario y mail 🡪 elige nivel de acceso (write|| read)

* + - ¿Qué es un repositorio público en GitHub?

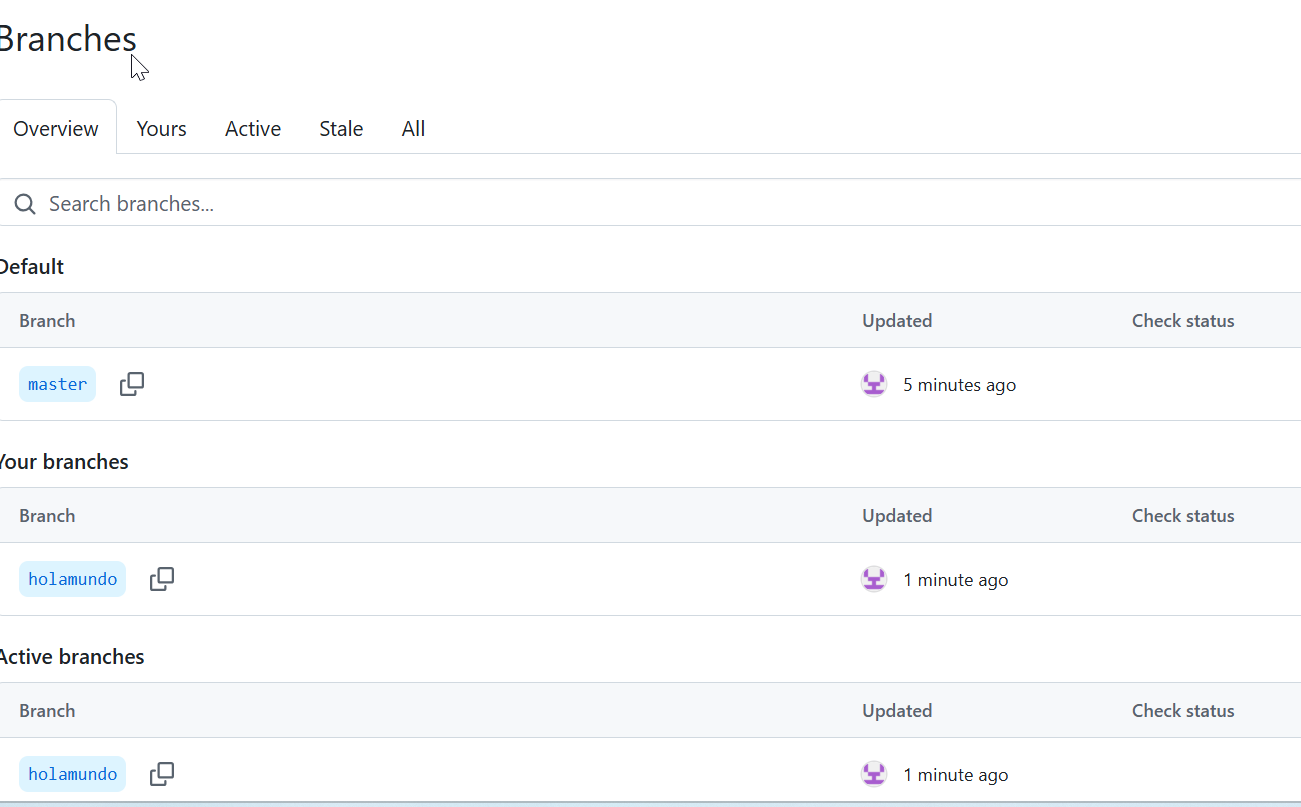
Significa que todos lo pueden ver

* + - ¿Cómo crear un repositorio público en GitHub?
* Hacé clic en el **"+"** en la esquina superior derecha.
* Elegí **"New repository"**.
* Escoger publico
  1. 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).

<https://github.com/J-Faruk-MP/Repositorio-Tarea2-Jhonny-Merrys>



2

* + - Creando Branchs
* Crear una Branch
* Realizar cambios o agregar un archivo
* Subir la Branch
* 
  1. 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

3

* 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

4

* 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.

5