**TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN**



**A DISTANCIA**

**Práctico 2: Git y GitHub**

**Estudiante: Avilas Martín Jonás**

# Actividades

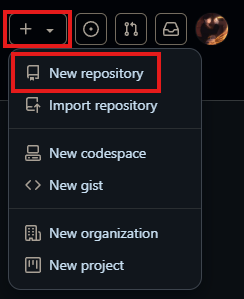
# 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

* ¿Qué es GitHub?

GitHub es una plataforma digital que facilita el almacenamiento y administración de proyectos de código mediante Git. Opera como un backup colaborativo y remoto en el que los programadores tienen la posibilidad de almacenar versiones de su código (mediante Git), colaborar con otros de manera ordenada y compartir sus proyectos de forma pública (código abierto) o privada.

* ¿Cómo crear un repositorio en GitHub?

1. Entrá a GitHub.
2. Arriba a la derecha hay un botón con un “+”.
3. Hacé clic ahí y elegí **“New repository”**.



* ¿Cómo crear una rama en Git?

git Branch nombre-rama

* ¿Cómo cambiar a una rama en Git?

git checkout rama

* ¿Cómo fusionar ramas en Git?

git merge rama

* ¿Cómo crear un commit en Git?

git commit -m "mensaje”

* ¿Cómo enviar un commit a GitHub?

git push

* ¿Qué es un repositorio remoto?

Es una copia del proyecto que está alojada en Internet, normalmente en plataformas como GitHub o GitLab.

* ¿Cómo agregar un repositorio remoto a Git?

Cuando tenés tu proyecto local (una carpeta) y querés vincularlo con GitHub:

git remote add origin https://github.com/tu-usuario/tu-repo.git

origin es el nombre que le das al remoto (la mayoría de las veces, se llama así)

Ese link es el del repositorio en GitHub

* ¿Cómo empujar cambios a un repositorio remoto?

git push origin main

* ¿Cómo tirar de cambios de un repositorio remoto?

git pull origin main

* ¿Qué es un fork de repositorio?

Un fork es como hacer una copia de un proyecto de otra persona a tu cuenta de GitHub, para poder modificarlo sin afectar el original.

* ¿Cómo crear un fork de un repositorio?

1. Entrás al repositorio que querés forkar.
2. Arriba a la derecha hay un botón que dice "Fork".
3. Hacés clic, y GitHub crea una copia en tu cuenta.

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

1. Hacés un fork del repo y clonás tu copia.
2. Creamos una rama nueva, hacemos cambios y los subimos (git push origin rama-nueva)
3. Entrás al repo en GitHub y le das a "Compare & pull request"
4. Completás título, descripción y le das a "Create pull request"

* ¿Cómo aceptar una solicitud de extracción?

En la pestaña "Pull Requests":

1. Revisar los cambios propuestos
2. Hacer clic en "Merge pull request"
3. Confirmar con "Confirm merge"

* ¿Qué es una etiqueta en Git?

Una etiqueta es como un marcador en el tiempo. Se usa para marcar versiones importantes del proyecto, por ej: "v1.0".

* ¿Cómo crear una etiqueta en Git?

git tag v1.0 (manera simple)

git tag -a v1.0 -m "Versión 1.0 estable" (manera más completa)

* ¿Cómo enviar una etiqueta a GitHub?

git push origin v1.0 (si sólo tenés una para enviar)

git push origin –tags (para subir directamente todas)

* ¿Qué es un historial de Git?

Es la lista de todos los commits hechos en un repositorio. Podés ver qué cambios se hicieron, cuándo y por quién.

* ¿Cómo ver el historial de Git?

git log

* ¿Cómo buscar en el historial de Git?

git log --grep="palabra clave" (por mensaje)

git log nombre-de-archivo.txt (Por archivo)

* ¿Cómo borrar el historial de Git?

rm -rf .git

* ¿Qué es un repositorio privado en GitHub?

Es un repo que solo vos y las personas que invites pueden ver o modificar. No aparece en las búsquedas públicas.

* ¿Cómo crear un repositorio privado en GitHub?

1. Entrás a GitHub y hacés clic en "New repository"
2. Completás nombre, descripción y tildás la opción "Private"
3. Crear

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

1. Entrás a tu repo
2. Vas a Settings > Collaborators
3. Ponés el nombre de usuario y hacés clic en "Add collaborator"

* ¿Qué es un repositorio público en GitHub?

Es un repo que cualquiera puede ver, descargar o clonar. Ideal para proyectos de código abierto o para compartir en un portfolio.

* ¿Cómo crear un repositorio público en GitHub?

1. En GitHub > New repository
2. Completás los datos y tildás "Public"
3. Crear

* ¿Cómo compartir un repositorio público en GitHub?

Copiás el link del repo y se lo pasás a quien quieras.

# 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/314-thon/Tp2-Git-GitHub>

# 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