**Práctico 2: Git y GitHub**

Alumno: Castellini Gonzalo

### ****¿Qué es GitHub?****

GitHub es una plataforma basada en la nube que permite a los desarrolladores gestionar y compartir código mediante el sistema de control de versiones Git. Permite a los equipos colaborar en proyectos de software, realizar seguimiento de cambios, gestionar versiones y coordinar desarrollos mediante herramientas como repositorios, ramas, pull requests y revisiones de código.

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

1. Inicia sesión en GitHub
2. Haz clic en el botón "+" en la esquina superior derecha y selecciona "New repository".
3. Introduce un nombre para el repositorio y una descripción (opcional).
4. Selecciona si será público o privado.
5. Opcionalmente, puedes inicializarlo con un archivo README, una licencia y un archivo .gitignore.
6. Haz clic en "Create repository".

### ****¿Cómo crear una rama en Git?****

En Git, una rama es una versión paralela del código que permite trabajar en nuevas funciones sin afectar la rama principal. Para crear una rama, usa el comando:

**git branch nombre\_rama**

### ****¿Cómo cambiar a una rama en Git?****

Para cambiar a una rama existente, usa:

**git checkout nombre\_rama**

### ****¿Cómo fusionar ramas en Git?****

Para combinar los cambios de una rama en otra (por ejemplo, unir una nueva función con la rama principal), sigue estos pasos:

1. Cambia a la rama en la que quieres fusionar los cambios (por ejemplo, main):

**git checkout main**

1. Fusiona la otra rama:

**git merge nombre\_rama**

1. Si hay conflictos, resuélvelos manualmente, dejando solamente el código valido y confirma los cambios.

### ****¿Cómo crear un commit en Git?****

Un commit es un punto de guardado en el historial de Git. Para hacer un commit:

1. Añade los archivos modificados al área de preparación:

**git add .**

1. Crea el commit con un mensaje descriptivo:

**git commit -m "Descripción del cambio"**

### ****¿Cómo enviar un commit a GitHub?****

Después de hacer un commit, envíalo al repositorio remoto en GitHub con:

**git push origin nombre\_rama**

### ****¿Qué es un repositorio remoto?****

Es una versión de un repositorio almacenada en un servidor en la nube, que permite a varios desarrolladores colaborar en un proyecto.

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

Si tienes un repositorio local y quieres asociarlo con GitHub, usa:

**git remote add origin URL\_del\_repositorio**

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

Para enviar los cambios al servidor remoto:

**git push origin nombre\_rama**

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

Si necesitas actualizar tu repositorio local con los cambios del remoto, usa:

**git pull origin nombre\_rama**

### ****¿Qué es un fork de repositorio?****

Un fork es una copia de un repositorio alojado en GitHub dentro de tu propia cuenta, que te permite realizar modificaciones sin afectar el original.

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

1. Ve al repositorio en GitHub.
2. Haz clic en el botón "Fork" en la esquina superior derecha.
3. GitHub creará una copia en tu cuenta.

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

1. En GitHub, ve a la pestaña "Pull requests".
2. Haz clic en "New pull request".
3. Selecciona la rama de tu fork y la rama en el repositorio original.
4. Añade un comentario explicando los cambios y envía la solicitud.

### ****¿Cómo aceptar una solicitud de extracción?****

1. Abre la pestaña "Pull requests" en GitHub.
2. Revisa los cambios propuestos.
3. Si todo está correcto, haz clic en "Merge pull request".
4. Confirma la fusión con "Confirm merge".

### ****¿Qué es una etiqueta en Git?****

Una etiqueta (tag) es una referencia a un estado específico de un repositorio, comúnmente usada para marcar versiones estables.

### ****¿Cómo crear una etiqueta en Git?****

****git tag <nombre-de-la-etiqueta>****

**Ejemplo**

**git tag -a v1.0 -m "Versión 1.0"**

### ****¿Cómo enviar una etiqueta a GitHub?****

**git push origin v1.0**

### ****¿Qué es un historial de Git?****

El historial de Git es el registro de todos los commits realizados en un repositorio, permitiendo ver los cambios y su autoría.

### ****¿Cómo ver el historial de Git?****

**git log**

Para una vista más compacta:

**git log --oneline**

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

**Por mensaje de commit**

**git log --grep="texto"**

**Por autor**

**git log --author="nombre"**

**Por fecha**

**git log --since="1 week ago" --until="today"**

**Buscar cambios en código**

**git log -S "texto" # En diffs**

**git grep "texto" # En código actual**

**Ver ramas y merges**

**git log --oneline --graph --all**

**Buscar en un commit específico**

**git show abc123**

### ****¿Cómo borrar el historial de Git?****

No se puede borrar completamente, pero se puede reescribir con:

# Rebase interactivo (ej., borrar los últimos n commits)

**git rebase -i HEAD~n**

# En el editor, marca con 'drop' los commits a eliminar.

# Forzar push después del rebase

**git push -f origin main**

Para eliminar todo el historial en un repositorio local, podrías usar:

**rm -rf .git**

**git init**

Pero esto no afectará al historial en GitHub.

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

Es un repositorio visible solo para los usuarios con permiso, ideal para proyectos privados o en desarrollo.

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

Al crear un repositorio en GitHub, selecciona la opción "Private".

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

1. Ve a la pestaña "Settings" del repositorio.
2. En "Manage access", haz clic en "Invite a collaborator".
3. Escribe el nombre de usuario y envía la invitación.

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

Un repositorio accesible para cualquier usuario de GitHub, útil para proyectos de código abierto.

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

Durante la creación del repositorio, selecciona la opción "Public".

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

Comparte la URL del repositorio con otros usuarios o agrégala en documentación, redes sociales o tu portafolio.

**2) Realizar la siguiente actividad:**

**Pongo a disposición en enlace al repositorio que se creo con el ejercicio:**

**https://github.com/gonzilla2021/TP2Prog1-TUPaD.git**

• Crear un repositorio.

o Dale un nombre al repositorio.

o Elije el repositorio sea público.

o Inicializa el repositorio con un archivo.

• Agregando un Archivo

o Crea un archivo simple, por ejemplo, "mi-archivo.txt".

**echo "Hola, Git!" > mi-archivo.txt**

o Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt"

en la línea de comandos.

**git add .**

**git commit -m "Agregando mi-archivo.txt"**

o Sube los cambios al repositorio en GitHub con git push origin main (o el

nombre de la rama correspondiente).

**git push origin main**

• Creando Branchs

o Crear una Branch

**git checkout -b nueva-funcion**

o Realizar cambios o agregar un archivo

o Subir la Branch

**git add .**

**git commit -m "Cambios en nueva-funcion"**

**git push origin nueva-funcion**

**3) Realizar la siguiente actividad:**

**Copio el enlace al repositorio que se creo con la solucion al ejercicio:**

[**https://github.com/gonzilla2021/conflict-exercise.git**](https://github.com/gonzilla2021/conflict-exercise.git)

**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, añgitadiendo 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.