TP N°2 Git y GitHub

Rojas Maximiliano

Soluciones.
Ejercicio 1:

- GitHub es una plataforma donde los programadores-desarrolladores puede publicar y compartir código, algo así como una red social de código. Utiliza Git para el control de versiones.
- Para crear un repositorio ingresamos a nuestro perfil de GitHub:
    - Hacemos clic en el **+** y seleccionamos **"New Repository".**
    - Nombramos el nuevo repositorio y agregamos una descripción (opcional pero recomendado)
    - Seleccionamos el tipo de visibilidad (Pública o privada).
    - Finalmente hacemos clic en **"Create repository"**.
- Para crear una rama en Git:
    - Ingresamos a la terminal y dentro del proyecto usamos el comando **git Branch nombre-de-la-rama**
- Para cambiar de rama lo hacemos con el comando **git checkout nombre-de-la-rama.**
- Para fusionar ramas nos movemos a la rama principal con **git checkout nombre-de-la-rama** y luego con **git merge nombre-otra-rama** fusionamos la segunda rama a la principal.
- Para crear un commit: usamos el comando **git add .** para decirle a Git que queremos subir algo. Para crear el commit usamos el comando **git commit -m "…"** (entre comillas podemos colocar un mensaje por si trata de algo nuevo o alguna modificación).
- Para subir el commit a GitHub usamos el comando **git push origin main.**
- Un repositorio remoto es una copia de nuestro proyecto que se encuentra en internet y a la cual otras personas pueden acceder y realizar cambios si lo desearan.
- Para agregar un repositorio remoto a Git usamos el comando **git remote add origin -URLDelProyecto-.**
- Para empujar cambios al repositorio remoto usamos el comando **git push origin main (**o el nombre de la rama).
- Para tirar de cambios de un repositorio remoto usamos el comando **git pull origin main.** Con esta instrucción incorporamos al repositorio local los cambios del repositorio remoto.
- Un **fork** de un repositorio es una copia de otro repositorio que se agrega en nuestra cuenta de GitHub. Se puede realizar cambios en el sin afectar al repositorio original.
- Para crear un **fork** nos trasladamos al repositorio que deseamos copiar y hacemos clic en el botón **Fork**.
- Para enviar una solicitud de extracción (pull request) a un repositorio debe seguir los siguientes pasos:
    - Realizamos un fork del repositorio indicado y luego subirlo de manera local.
    - Realizamos los cambios deseados y hacemos un git commit (-m "…" si deseamos agregarle un título).
    - Luego lo subimos a GitHub con el comando git push.
    - Por último, en el repositorio original hacemos clic en la opción **"créate pull request".**
- Para aceptar la solicitud de extracción debemos dirigirnos al repositorio en GitHub, ver los **pull requests** sugeridos y si estamos de acuerdo aceptarlos haciendo clic en **"Merge pull request".**

- Una etiqueta en Git es una referencia en el historial del repositorio para marcar algo importante.
- Para crear una etiqueta usamos el comando **git tag -a nombre-de-la-etiqueta -m "mensaje".**
- Para subir estas etiquetas a GitHub debemos usar el comando **git push origin - -tags.**
- El historial de Git es una lista donde están los cambios realizados en un repositorio, allí se pueden consultar los commits realizados.
- Podemos ver el historial de Git usando el comando **git log**.
- Podemos buscar dentro del historial utilizando el comando **git log -p** (y podemos agregar **-3** por ejemplo para referirnos a los últimos 3 commits).
- Para borrar el historial usamos el comando **git reset.**
- Un repositorio privado en GitHub solo puede ser visto por el creador y por quienes tengan permiso para acceder a él.
- Para crear un repositorio privado debemos seguir los pasos comentados más arriba con la diferencia que debemos seleccionar "privado" en la visibilidad de este.
    - *Hacemos clic en el **+** y seleccionamos **"New Repository".***
    - *Nombramos el nuevo repositorio y agregamos una descripción (opcional pero recomendado)*
    - *Seleccionamos el tipo de visibilidad (Pública o privada).*
    - *Finalmente hacemos clic en **"Create repository"**.*
- Para invitar a alguien a un repositorio privado en GitHub seguimos los siguientes pasos.
    - Vamos al repositorio que deseamos compartir y en configuración seleccionamos "colaboradores".
    - Ingresamos el nombre de usuario o email de la persona y hacemos clic en "agregar colaborador".
- Un repositorio público en GitHub a diferencia del privado que mencionábamos antes es un repositorio al cual puede ingresar cualquier persona. Ideal para mostrar proyectos y sugerirlos a posibles colaboraciones o bien para los mostrar los trabajos ya realizados como un porfolio.
- Para crear un repositorio público en GitHub debemos seguir los siguientes pasos:
    - *Hacemos clic en el **+** y seleccionamos **"New Repository".***
    - *Nombramos el nuevo repositorio y agregamos una descripción (opcional pero recomendado)*
    - *Seleccionamos el tipo de visibilidad (Pública o privada).*
      *Finalmente hacemos clic en **"Create repository"**.*
- Para compartir el repositorio público nos dirigimos al repositorio deseado, en el botón "Code" desplegamos y copiamos la URL del repositorio; luego podemos compartir el enlace por el medio que deseemos.

Ejercicio 2.

https://github.com/Yamil-Rojas/EjercicioTP.git

Ejercicio 3.

https://github.com/Yamil-Rojas/conflict-exercise.git

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?
- ¿Cómo crear un repositorio en GitHub?
- ¿Cómo crear una rama en Git?
- ¿Cómo cambiar a una rama en Git?
- ¿Cómo fusionar ramas en Git?
- ¿Cómo crear un commit en Git?
- ¿Cómo enviar un commit a GitHub?
- ¿Qué es un repositorio remoto?
- ¿Cómo agregar un repositorio remoto a Git?
- ¿Cómo empujar cambios a un repositorio remoto?
- ¿Cómo tirar de cambios de un repositorio remoto?
- ¿Qué es un fork de repositorio?
- ¿Cómo crear un fork de un repositorio?

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
- ¿Cómo aceptar una solicitud de extracción?
- ¿Qué es un etiqueta en Git?
- ¿Cómo crear una etiqueta en Git?
- ¿Cómo enviar una etiqueta a GitHub?
- ¿Qué es un historial de Git?
- ¿Cómo ver el historial de Git?
- ¿Cómo buscar en el historial de Git?
    - ¿Cómo borrar el historial de Git?
- ¿Qué es un repositorio privado en GitHub?
- ¿Cómo crear un repositorio privado en GitHub?
- ¿Cómo invitar a alguien a un repositorio privado en GitHub?
- ¿Qué es un repositorio público en GitHub?
- ¿Cómo crear un repositorio público en GitHub?
- ¿Cómo compartir un repositorio público en GitHub?

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

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.