Nombre: Diego Rual Montes

Este es mi Repositorio que contiene la Actividad 2: https://github.com/UshioVII/TPS-UTN.git

Repositorio remoto de la Actividad 3: https://github.com/UshioVII/Actividad3conflict-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?

GitHub es una plataforma que permite almacenar, gestionar y compartir código utilizando Git. Básicamente, sirve para:

- Guardar proyectos en código en un repositorio.
- Colaborar con otras personas, permitiendo que varias personas trabajen en el mismo proyecto sin sobre escribir cambios.
- Hacer seguimiento de versiones, viendo qué cambios se hicieron, cuándo y por quién.
- Publicar proyectos para que otros los usen o contribuyan.

- ¿Cómo crear un repositorio en GitHub?

1. Inicia sesión en GitHub.

2. En la parte superior derecha, haz clic en el botón **"New"** o en el ícono de **"+"** y selecciona **"New repository"**.
3. Escribe un **nombre** para tu repositorio.
4. Opcionalmente, agrega una **descripción**.
5. Elige si quieres que tu repositorio sea **público** (visible para todos) o **privado** (solo para ti y las personas que invites).
6. Puedes marcar la opción **"Initialize this repository with a README"** si deseas que incluya un archivo inicial.
7. Finalmente, haz clic en **"Create repository"**.

¡¡PASO IMPORTANTE!!
Antes de continuar y subir nuestros archivos, necesitamos asegurarnos de que Git está configurado con nuestro usuario de GitHub.

0. Abrimos la terminal de Git y configuramos nuestro usuario de GitHub y email o Gmail

```
git config --global user.name "TuNombreDeUsuario"
git config --global user.email "tuemail@ejemplo.com"
```

Por si tienes Gmail este es el otro comando:

```
git config --global user.gmail "tugmail@gmail.com"
```

Después de haber conectado nuestro Git con nuestro GitHub, podemos seguir-

1. **Abre la terminal** y navega a la carpeta de tu proyecto.
   Por Ejemplo: cd Download/Projectos/tp1-codigo

2. Ejecuta los siguientes comandos:

```
git init  # Inicializa un repositorio Git en la carpeta
git add .  # Agrega todos los archivos al repositorio
git commit -m "Primer commit"  # Guarda los cambios con un mensaje
git branch -M main  # Asegura que la rama principal se llame 'main' Esto varia con ‘master’..
```

Asegúrate de que tu rama este escrita correctamente.

```
git remote add origin https://github.com/tu-usuario/tu-repositorio.git  # Conecta con GitHub
git push -u origin main  # Sube los archivos al repositorio
```

- ¿Cómo crear una rama en Git?

  Para crear una nueva rama en nuestro proyecto, vamos escribir en nuestro projecto:
  git branch nombre-de-la-rama

- ¿Cómo cambiar a una rama en Git?

  Vamos a lograr eso con:
  git checkout nombre-de-la-rama

- ¿Cómo fusionar ramas en Git?
  Tuviéramos que posicionarnos la rama actual para después funcionar, con rama2.

  git merge rama

- ¿Cómo crear un commit en Git?
  Con el comando commit Confirmariamos los cambios de nuestro proyecto:

  git commit -m "mensaje"

- ¿Cómo enviar un commit a GitHub?

  Primero confirmamos los archivos modificados:

  git status

  Despues vamos agregar los archivos con:

  git add .

  O si queremos que sea mas preciso podrías usar:

  git add nombre-archivo/

  Despues hacemos un commit para darle una descripción a nuestros cambios:

  git commit -m “Archivos agregados y más archivos agregados”

  Y por ultimo subimos todo a los cambios al repositorio:

  git push origin master # Ojo podría ser que estemos trabajando en otra rama, así que hay que verificar si la rama en la que estamos trabajando es la correcta.

- ¿Qué es un repositorio remoto?

  Un repositorio remoto es una versión del proyecto almacenada en la nube, como lo es GitHub. También Permite que varias personas trabajen en el código y mantengan sincronizados sus cambios.

- ¿Cómo agregar un repositorio remoto a Git?

  git remote add origin https://github.com/TuUsuario/TuRepositorio.git

- ¿Cómo empujar cambios a un repositorio remoto?

  Tenes dos formas una es empujando todo con

  git add .

  Y esta forma seria mas controlada:

  git add nombre-del-archivo.py

- ¿Cómo tirar de cambios de un repositorio remoto?

  Para **tirar cambios de un repositorio remoto** sigue estos pasos:

1. **Verifica la rama actual** donde estás trabajando usando git branch. Esto te muestra la rama en la que estás y si necesitas cambiar a otra rama antes de hacer pull.
2. **Haz pull de los cambios** del repositorio remoto con el comando git pull origin [nombre_de_rama]. Por ejemplo, si estás trabajando con la rama principal, usa git pull origin main, o si estás en otra rama, como dev, usa git pull origin dev.
3. Si **hay conflictos** entre los cambios del remoto y los tuyos, Git te avisará. Abre los archivos con conflictos, resuélvelos manualmente, y luego marca los archivos como resueltos con git add ..
4. **Haz un commit** para registrar la resolución de los conflictos con git commit -m "Resolviendo conflictos".
5. Finalmente, **verifica el estado de tu repositorio** con git status para asegurarte de que todo está en orden.

- ¿Qué es un fork de repositorio?

  Un **fork** es una copia personal de un repositorio que se hace en plataformas como **GitHub**. Permite trabajar de manera independiente sobre el código de otro repositorio sin afectar al original.

- ¿Cómo crear un fork de un repositorio?

  1. Entra al repositorio que deseas **forkear** en GitHub.
  2. Haz clic en el botón **"Fork"** en la parte superior derecha de la página.
  3. Se creará una copia del repositorio en tu cuenta de GitHub, donde podrás trabajar libremente.

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

  Para enviar una pull request a un repositorio, lo primero que hago es hacer un **fork** del repositorio original. Esto lo hago porque, como no tengo acceso directo para hacer cambios en el repositorio principal, hago una copia del proyecto en mi cuenta. Entonces, en GitHub, busco el repositorio al que quiero contribuir y hago clic en **Fork**.
  Después, una vez que tengo mi copia en mi cuenta, clono ese repositorio a mi pc para poder trabajar en él. Para eso, copio la URL del repositorio en GitHub, abro la terminal y escribo:

  git clone <URL del repositorio>

  Ya con el repositorio en mi computadora, lo siguiente es crear una **nueva rama** para poder trabajar en mis cambios sin afectar la rama principal. Para esto, escribo:

  git checkout -b nombre-de-la-rama

  Hago los cambios que necesite en el código, luego los **agrego** al área de preparación con:

  git add .

  Y **commit** para guardar esos cambios de manera que se sepa qué hice:

  git commit -m "Descripción de los cambios"

  Una vez que tengo mis cambios listos, los **empujo** a mi repositorio en GitHub:

  git push origin nombre-de-la-rama

- ¿Cómo aceptar una solicitud de extracción?

1. Revisa la Pull Request: Ve a la pestaña Pull requests en el repositorio y haz clic en la solicitud que quieres revisar.
2. Verifica los cambios: Revisa el código propuesto y asegúrate de que no haya conflictos o errores.
3. Haz el Merge: Si todo está bien, haz clic en Merge pull request y luego confirma con Confirm merge.
4. Elimina la Rama (opcional): Si ya no necesitas la rama de la pull request, puedes eliminarla con Delete branch.

- ¿Qué es un etiqueta en Git?

  Una **etiqueta** (o *tag*) en Git es una referencia fija a un punto específico en la historia de un repositorio, generalmente usada para marcar versiones o hitos importantes del proyecto. A diferencia de las ramas, que son referencias móviles, las etiquetas son inmutables una vez que se crean, lo que las hace útiles para señalar versiones de lanzamiento (como *v1.0*, *v2.0*).

- ¿Cómo crear una etiqueta en Git?

  git tag nombre # Crea un tag
  git tag -a nombre -m mensaje # Crea un tag con mensaje
  git tag -a nombre hash -m mensaje # Crea un tag en commit específico

- ¿Cómo enviar una etiqueta a GitHub?

  Ejemplo básico:

  Para ver todas las etiquetas:

  git tag

  Para crear una etiqueta con una anotación:

  git tag -a v0.010 -m "Alpha 0.010"

  Este comando es para empujar una tequita al repositorio remoto

  git push origin v0.010

- ¿Qué es un historial de Git?

  El **historial de Git** es un registro de todos los cambios realizados en un repositorio de Git a lo largo del tiempo. Cada vez que realizas un **commit**, Git guarda un punto en el tiempo que incluye el código, los mensajes de los commits, las diferencias respecto al estado anterior, y otros metadatos (como el autor y la fecha del cambio).

- ¿Cómo ver el historial de Git?

  Con el comando: git log Muestra una lista de los commits en el repositorio, incluyendo el hash, autor, fecha y mensaje del commit.

  Y con este otro comando: git log --oneline Muestra los commits en una sola línea con el hash abreviado y el mensaje.

- ¿Cómo buscar en el historial de Git?

  Podríamos usar el `git log`, para buscar en el historial, pero si somos mas vivos podemos agregar opciones y filtros `git log --decorate --all --graph --oneline`

  El comando git log --decorate --all --graph --oneline es muy útil para obtener una visualización clara y concisa del historial de tu repositorio Git.

  Ejemplo visual:

```
* 123abc4 (HEAD -> main, origin/main) Commit más reciente
* 567def8 Segunda versión de la funcionalidad
| * 890ghi (feature-branch) Feature en desarrollo
|/
* 234jkl7 Commit en la rama principal
```

  Explicación del ejemplo:

  - Cada asterisco (*) representa un commit.
  - El gráfico de barras (|) y las líneas diagonales (/) indican la estructura de las ramas.
  - Las referencias entre paréntesis, como (HEAD -> main, origin/main), muestran en qué ramas están los commits. HEAD indica en qué rama estás actualmente.
  - Los mensajes de commit están a la derecha de los hashes de los commits.

• ¿Cómo borrar el historial de Git?

Si quieres eliminar todo el historial de commits pero mantener los archivos en el estado actual, sigue estos pasos:

```
rm -rf .git  # Elimina la carpeta oculta de Git (¡Se pierde todo el historial!)
git init     # Inicializa un nuevo repositorio vacío
git add .    # Agrega todos los archivos al nuevo repositorio
git commit -m "Reinicio del historial"
git remote add origin <URL_DEL_REPOSITORIO>  # Vuelve a vincular el remoto
git push -f origin master  # Fuerza el push para sobrescribir el historial remoto
```

Si quieres eliminar solo algunos commits específicos, usa un **rebase:**

```
git rebase -i HEAD~N  # Reemplaza N con la cantidad de commits a modificar
```

Esto abrirá un editor de texto donde podrás elegir qué commits eliminar (drop).
Luego, actualiza el repositorio remoto con:

```
git push -f origin master  # Fuerza la actualización del historial
```

• ¿Qué es un repositorio privado en GitHub?

Un **repositorio privado** en GitHub es un repositorio en el que el acceso y visibilidad del código fuente y otros archivos almacenados están restringidos solo a las personas que tú decidas invitar o permitir. A diferencia de un repositorio público, que es accesible para cualquier persona en Internet, un repositorio privado está oculto y solo los usuarios autorizados pueden verlo, clonarlo o contribuir en él.

• ¿Cómo crear un repositorio privado en GitHub?

- En GitHub, ve a tu perfil y haz clic en "New" o "Nuevo" para crear un repositorio.
- En la pantalla de configuración del repositorio, marca la opción **"Private"** para que el repositorio sea privado.

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

- Una vez que el repositorio está creado, puedes invitar a colaboradores en la sección de **Settings** > **Manage access** (Configuración > Administrar acceso).
- Desde ahí, puedes agregar personas con su nombre de usuario o correo electrónico, y asignarles permisos para ver o modificar el repositorio.

• ¿Qué es un repositorio público en GitHub?

Un **repositorio público** en GitHub es un repositorio cuyo contenido está accesible para cualquier persona en Internet. Cualquier usuario de GitHub (o incluso personas que no tienen cuenta en GitHub) pueden ver, clonar, bifurcar (fork), y contribuir a este repositorio, dependiendo de los permisos configurados. Los repositorios públicos son ideales para compartir tu código con la comunidad y permitir que otros contribuyan, lo que es muy común en proyectos de código abierto.

• ¿Cómo crear un repositorio público en GitHub?

Crear un nuevo repositorio:

- Inicia sesión en GitHub y ve a tu perfil.
- Haz clic en "New" o "Nuevo" para crear un repositorio.
- En la configuración del repositorio, asegúrate de seleccionar la opción **"Public"** (Público).
- Completa los demás detalles, como el nombre del repositorio, la descripción y si quieres incluir un archivo README, una licencia, etc.
- Haz clic en "Create repository" para crear el repositorio.

• ¿Cómo compartir un repositorio público en GitHub?

Para compartir un **repositorio público en GitHub**, simplemente necesitas obtener la URL del repositorio y compartirla con las personas a través del medio que prefieras (correo electrónico, redes sociales, etc.)

2) Realizar la siguiente actividad:

• 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".
o Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.
o Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).

• Creando Branchs

o Crear una Branch
o Realizar cambios o agregar un archivo

o 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.