TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

Práctico 2: Git y GitHub

Estudiante: Fulladoza, Pablo Facundo
Comisión: 2025-13

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 donde se puede almacenar y compartir Repositorios de forma Remota.
- ¿Cómo crear un repositorio en GitHub?
  Para crear un repositorio en GitHub ingresar a tu usuario, ir a repositorios y apretar Nuevo, una vez asignado el nombre y unas breves configuraciones el mismo ya se encuentra disponible para su uso.
- ¿Cómo crear una rama en Git?
  Para crear una rama o branch en Git se utiliza el comando “git branch + <nombreDeRama>”.
- ¿Cómo cambiar a una rama en Git?
  Se utiliza el comando “git checkout + <nombreDeRama>”
- ¿Cómo fusionar ramas en Git?
  Se utiliza el comando “git merge + <nombreDeRama>” para fusionar la rama actual con la indicada en el comando.
- ¿Cómo crear un commit en Git?
  Para esto se utiliza el comando “git commit”.

- ¿Cómo enviar un commit a GitHub?
  Se realiza el commit de forma local y luego se realiza un git push.
- ¿Qué es un repositorio remoto?
  Un repositorio remoto es un repositorio ubicado en internet/nube, lo cual nos permite acceder a el sin tenerlo alojado en nuestra pc.
- ¿Cómo agregar un repositorio remoto a Git?
  Se utiliza el comando “git remote add + <nombreRepositorio> <urlRepositorio>”
- ¿Cómo empujar cambios a un repositorio remoto?
  Se utiliza el comando “git push + <nombreRepositorio>”
- ¿Cómo tirar de cambios de un repositorio remoto?
  En caso de que se requiera revertir los cambios eliminado el último commit se debe cambiar el head al commit anterior y luego realizar un push al repositorio remoto.
- ¿Qué es un fork de repositorio?
  Un fork es una copia de otro repositorio que se realiza para poder trabajar con este sin afectar el repositorio original.
- ¿Cómo crear un fork de un repositorio?
  Para crear un fork de un repositorio en GitHub simplemente se accede al repositorio y se presiona el botón de fork.
- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
  En el repositorio se encuentra un botón de “Contribute” donde se puede abrir un pull request.
- ¿Cómo aceptar una solicitud de extracción?
  En el menú de pull request de nuestro perfil de github verificamos los cambios y los aceptamos.
- ¿Qué es un etiqueta en Git?
  Se utiliza para etiquetar un commit, esta etiqueta tiene un nombre, fecha y usuario que la etiqueto y al ser mostrada trae el commit donde se encuentra.
- ¿Cómo crear una etiqueta en Git?
  Se utiliza el comando “git tag <nombreEtiqueta>”
- ¿Cómo enviar una etiqueta a GitHub?
  La etiqueta se envía junto con el commit al utilizar el comando push.
- ¿Qué es un historial de Git?
  El historial de git posee todos los commits realizados sobre un repositorio.
- ¿Cómo ver el historial de Git?
  Se utiliza el comando “git log”
- ¿Cómo buscar en el historial de Git?
  Se puede buscar utilizando comandos adicionales como “--grep” para buscar en el mensaje, “--author” para buscar según quien realizo el commit, “--<nombreArchivo>” para ver solo aquellos que afectaron al archivo indicado, etc.
- ¿Cómo borrar el historial de Git?
  Se debe crear una nueva rama temporal, se agregan todos los archivos y se realiza un commit. Luego se borra la rama anterior y se renombra la rama temporal.

Luego al realizar el push en Github se encontrará un solo commit.

- ¿Qué es un repositorio privado en GitHub?
  Un repositorio privado en Github es un repositorio que no puede ser visto por otros usuarios si no se comparte explicitamente.
- ¿Cómo crear un repositorio privado en GitHub?
  Se crea al igual que un repositorio común, pero se selecciona privado en vez de público.
- ¿Cómo invitar a alguien a un repositorio privado en GitHub?
  En la configuración del repositorio se ingresa a “Collaborators” y ahí se los puede invitar utilizando usuario, mail, etc.
- ¿Qué es un repositorio público en GitHub?
  Un repositorio público es un repositorio que se comparte con todos los usuarios de la plataforma.
- ¿Cómo crear un repositorio público en GitHub?
  Al crear un repositorio se indica que el mismo es público.
- ¿Cómo compartir un repositorio público en GitHub?
  En estos casos no se necesita invitar a los otros usuarios, con facilitar el link del mismo estos ya podrían ingresar.

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.

Se realiza en: https://github.com/Fulla1996/ConflictPractice/commits/main/