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 en la nube donde puedes almacenar y compartir con otros usuarios código mediante repositorios, se puede llevar registro de los cambios que se van generando.

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

  En el margen superior izquierdo de la página de Github encontramos un botón que dice New. Al entrar en esta nueva ventana podemos poner le un nombre a nuestro nuevo repositorio , definir como público o privado, agregar un archivo README, finalmente haremos click en el botón create repository para finalizar.

- **¿Cómo crear una rama en Git?**

  para crear una nueva rama en la terminal de Visual studio escribimos el comando git branch seguido del nombre de la Rama.

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

  Para cambiar a una rama debo escribir en la terminal el comando git checkout seguido del nombre de la rama

- **¿Cómo fusionar ramas en Git?**

  Para fusionar ramas utilizo el comando git merge seguido del nombre la rama que

quiero fusionar, colocando previamente con git checkout en la rama donde quiero que quede fusionado mi código.

- **¿Cómo crear un commit en Git?**

  Creamos un commit escribiendo en la terminal el comando git Commit -m seguido de un mensaje que lo identifique

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

  Luego de hacer el commit debo ingresar el comando git push.

- **¿Qué es un repositorio remoto?**
  Es el lugar donde se almacena el código, los archivos y el historial de revisiones de un archivo. Los repositorios pueden contar con múltiples colaboradores y pueden ser públicos como privados.

- **¿Cómo agregar un repositorio remoto a Git?**
  Lo puedo hacer con el comando git remote add

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

  con el comando git push

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

  Con el comando git fetch , clone o pull depende que queremos hacer.

- **¿Qué es un fork de repositorio?**

  Fork es una copia de un repositorio creado en una cuenta diferente permitiendo desarrollar cambios, sin afectar el principal. A diferencia del clonado este se realiza generando una copia de la cuenta del usuario.

- **¿Cómo crear un fork de un repositorio?**
  Para crear un fork entró en el repositorio que quiero y hago click en el botón fork q se encuentra a la derecha de la pantalla

- **¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?**
  Elijo la rama donde quiero fusionar mi código, luego hago click en comare y pull request. comparo y elijo los cambios , luego le asignó un título y descripción a la solicitud. luego creamos la solicitud
- **¿Cómo aceptar una solicitud de extracción?**

  revisamos los cambios , realizamos algún comentario y luego lo aceptamos.
- **¿Qué es una etiqueta en Git?**

  Permite marcar puntos específicos del historial de un repositorio como importantes.
- **¿Cómo crear una etiqueta en Git?**

  se crea una etiqueta anotada mediante el comando `git tag -a`
- **¿Cómo enviar una etiqueta a GitHub?**

  lo puedo enviar haciendo el git push seguido del --tag
- **¿Qué es un historial de Git?**

  Son todos los cambios que se realizaron sobre un código en particular
- **¿Cómo ver el historial de Git?**

  puedo ver el historial mediante el comando git log , así puedo ver el historial en orden cronológico inverso.
- **¿Cómo buscar en el historial de Git?**
  se puede hacer agregando comandos a git log, como por ejemplo git log -p -2 nos muestra las últimas dos entradas

  - **¿Cómo borrar el historial de Git?**
    eliminando la carpeta .git
- **¿Qué es un repositorio privado en GitHub?**
  Es un repositorio que no tiene visibilidad al publico solo puede acceder la persona que lo crea y personas autorizadas.
- **¿Cómo crear un repositorio privado en GitHub?**

  Al crear el repositorio se debe elegir si crearlo publico o privado.
- **¿Cómo invitar a alguien a un repositorio privado en GitHub?**
  Debo agregarlos como colaboradores en mi repositorio
- **¿Qué es un repositorio público en GitHub?**
  Es un repositorio al que puede acceder cualquier persona sin solicitar ningún permiso
- **¿Cómo crear un repositorio público en GitHub?**
  Cuando cree el repositorio debe elegir la opción público .

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

  Puedo compartirlo mediante la direccion de Url.

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

link:https://github.com/Gistefani/Programacion-UTN

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.

Link: https://github.com/Gistefani/conflict-exercise