Actividad Práctica: Aplicación de Conceptos Intermedios de Git

Conceptos teóricos sobre Git a nivel intermedio, incluyendo el uso de ramas, fusión de cambios, resolución de conflictos, `git stash`, y buenas prácticas de commits y trabajo colaborativo.

Instrucciones Generales:

1. Trabajo en equipos de 3-4 personas.
2. Cada equipo creará un proyecto en un repositorio de GitHub (u otro servicio similar), y colaborarán para desarrollar nuevas características y resolver conflictos de fusión.
3. Se evaluará el uso adecuado de ramas, commits atómicos, resolución de conflictos, `git stash`, y la calidad de los mensajes de commit.

Pasos del Ejercicio:

Parte 1: Configuración del Repositorio

1. **Crear un repositorio remoto**:
    - Uno de los equipo debe crear un nuevo repositorio en GitHub.
    - Los demás miembros del equipo deben clonar este repositorio en sus máquinas locales.

      ```
      git clone https://github.com/usuario/proyecto.git
      ```

2. **Inicializar el repositorio**:
    - El equipo debe crear un archivo README.md en la rama principal (`main`) y escribir una descripción breve del proyecto.
    - Luego, deben hacer un commit y enviar (`push`) este cambio al repositorio remoto.

      ```
      git add README.md
      git commit -m "Añadido archivo README con descripción del proyecto"
      git push origin main
      ```

Parte 2: Trabajo con ramas y desarrollo colaborativo

1. **Crear nuevas ramas**:
    - Cada estudiante debe crear una rama para trabajar en una nueva característica del proyecto. Cada rama debe seguir la convención de nombres adecuada:
        - Ejemplo: `feature/nueva-funcionalidad` o `feature/agregar-estilos-css`.

      ```
      git branch feature/nueva-funcionalidad
      git checkout feature/nueva-funcionalidad
      ```

2. **Desarrollo en las ramas**:
    - Cada miembro debe modificar algún archivo (por ejemplo, crear una nueva página HTML, agregar una funcionalidad en JavaScript, o añadir estilos CSS) en su rama.
    - Hacer commits atómicos por cada pequeño cambio realizado, siguiendo las buenas prácticas.

      ```
      git add archivo.html
      git commit -m "Creada nueva página HTML con estructura básica"
      ```

3. **Enviar los cambios al repositorio remoto**:
    - Cada miembro debe enviar los cambios de su rama al repositorio remoto.

      ```
      git push origin feature/nueva-funcionalidad
      ```

Parte 3: Pull Requests y resolución de conflictos (20 minutos)

1. **Crear Pull Requests**:
    - Los miembros del equipo deben crear un Pull Request (PR) en GitHub para fusionar sus ramas de características con la rama principal (`main`).
    - Los compañeros del equipo deben revisar el PR antes de aprobar la fusión. Se pueden dejar comentarios sobre los cambios propuestos.
2. **Simular conflictos**:
    - Para simular un conflicto de fusión, dos de los integrantes deben modificar la misma sección de un archivo (por ejemplo, `index.html`), pero desde ramas diferentes.
    - Al intentar fusionar estas ramas en `main`, Git detectará un conflicto y los compañeros deberán resolverlo.
3. **Resolver conflictos de fusión**:
    - Uno de los del equipo debe fusionar su rama con `main` y luego, cuando otro intente hacer lo mismo, ocurrirá el conflicto.
    - El estudiante afectado debe resolver el conflicto manualmente en su máquina,
        - **Editar el archivo** para decidir qué cambios conservar.
        - **Añadir el archivo resuelto** al staging area:

          ```
          git add archivo-en-conflicto.html
          ```

        - **Confirmar la fusión** con un commit:

          ```
          git commit -m "Resuelto conflicto de fusión entre main y
          feature/nueva-funcionalidad"
          ```

Parte 4: Uso de `git stash`

1. **Simular una interrupción en el trabajo**:
    - Mientras se trabaja en una nueva funcionalidad en una rama, se pedirá que cambien a la rama principal (`main`) para resolver un problema urgente.
    - Los miembros deben guardar su trabajo actual sin hacer un commit, utilizando `git stash`:

      ```
      git stash
      ```

2. **Cambiar a la rama principal y actualizar**:
    - Cambiar a la rama principal para obtener la última versión del código:

      ```
      git checkout main
      git pull origin main
      ```

3. **Volver a la rama de trabajo y restaurar los cambios guardados**:
    - Después de resolver el problema en la rama principal, los integrantes del equipo volverán a su rama de trabajo y aplicarán los cambios guardados con `git stash apply`:

      ```
      git checkout feature/nueva-funcionalidad
      git stash apply
      ```

Parte 5: Finalización del proyecto y evaluación

1. **Fusionar todas las ramas en `main`**:
    - Después de completar todas las funcionalidades y resolver los conflictos, los estudiantes deben fusionar sus ramas con la rama principal.
    - El equipo debe verificar que todos los cambios han sido integrados correctamente en la rama principal.
2. **Revisión final del repositorio**:
    - Se revisará el historial de commits en el repositorio remoto para asegurarse de que se han seguido las buenas prácticas:
        - Commits atómicos.
        - Mensajes de commit claros y descriptivos.
        - Uso adecuado de ramas y Pull Requests.
        - Resolución correcta de conflictos.
