HERRAMIENTAS DEL DESARROLLO

Laboratorio: Trabajo Colaborativo con Sistema de Control de Versiones en la Nube

Objetivo:

El objetivo de este laboratorio es enseñar a los estudiantes cómo trabajar de manera colaborativa utilizando un sistema de control de versiones en la nube, específicamente con plataformas como GitHub o GitLab. Los estudiantes aprenderán cómo clonar, colaborar, gestionar ramas, realizar Pull Requests, y resolver conflictos de fusión en un entorno de equipo.

Procedimientos Básicos:

1. Crear una Cuenta en GitHub (o GitLab):

1. Dirígete a GitHub o GitLab y crea una cuenta si aún no tienes una.
2. Accede a tu cuenta para comenzar a trabajar con repositorios remotos.

2. Crear un Repositorio en la Nube:

1. En GitHub, haz clic en el botón de "New" para crear un nuevo repositorio.
2. Proporciona un nombre, una descripción y selecciona si deseas que el repositorio sea público o privado.
3. No es necesario añadir un archivo README ni un `.gitignore` en este momento.

3. Clonar el Repositorio Remoto en tu Máquina Local:

Para clonar el repositorio, copia la URL del repositorio desde GitHub:

```
git clone https://github.com/<usuario>/<nombre_repositorio>.git
```

4. Configurar un Archivo `.gitignore`:

Crea un archivo `.gitignore` en el directorio raíz del proyecto para indicar qué archivos o carpetas deben ser ignorados (por ejemplo, archivos temporales o dependencias).

```
node_modules/
*.log
```

5. Realizar un Commit Inicial:

Añade un archivo README que explique el propósito del proyecto y realiza un commit inicial:

```
git add README.md
git commit -m "Primer commit con archivo README"
git push origin main
```

6. Crear y Trabajar en una Nueva Rama:

Para trabajar en nuevas características sin afectar la rama principal (`main`), crea una nueva rama:

```
git checkout -b feature/nueva-caracteristica
```

7. Realizar Cambios y Hacer Commits:

Haz los cambios en el código y realiza commits periódicos para registrar el progreso de tus modificaciones.

```
git add <archivo_modificado>
git commit -m "Agregado nuevo método para la funcionalidad"
```

8. Subir los Cambios al Repositorio Remoto:

Para compartir tus cambios con el equipo, sube los commits de tu rama al repositorio remoto:

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

9. Crear una Pull Request (PR):

Para integrar tu rama `feature/nueva-caracteristica` a la rama `main`, crea una Pull Request:

1. Dirígete al repositorio en GitHub.
2. Haz clic en "New Pull Request".
3. Selecciona `feature/nueva-caracteristica` como la rama origen y `main` como la rama destino.
4. Agrega un comentario que explique los cambios realizados.

10. Revisión y Fusión de Pull Request:

Después de revisar la Pull Request, el líder del proyecto o un miembro del equipo puede fusionar la rama con la rama principal (`main`).

- Haz clic en el botón "Merge Pull Request" y luego en "Confirm Merge".

11. Resolver Conflictos de Fusión:

Si existen conflictos entre las ramas, Git no puede fusionarlas automáticamente. Debes resolverlos manualmente:

1. Git marcará el archivo con conflictos.
2. Edita el archivo para resolver los conflictos.

3. Agrega el archivo resuelto y realiza un commit:
4. `git add <archivo_resuelto>`
5. `git commit -m "Conflictos resueltos"`

12. Actualizar tu Repositorio Local con los Cambios Remotos:

Para mantener tu repositorio local actualizado con los cambios realizados por otros colaboradores, ejecuta:

```
git pull origin main
```

Preguntas :

1. **¿Qué es un sistema de control de versiones distribuido?**
   - **Respuesta**: Un sistema de control de versiones distribuido (DVCS) como Git permite a cada colaborador tener una copia completa del repositorio, incluyendo el historial de todos los cambios, lo que facilita trabajar sin conexión y realizar colaboraciones descentralizadas.
2. **¿Qué es una Pull Request (PR) y qué propósito cumple?**
   - **Respuesta**: Una Pull Request es una solicitud para fusionar una rama con otra, típicamente desde una rama de características a la rama principal (`main`). Permite revisar y discutir los cambios antes de que se integren al proyecto.
3. **¿Qué significa hacer un commit en Git?**
   - **Respuesta**: Un commit en Git es una operación que guarda los cambios realizados en los archivos de un proyecto. Cada commit está asociado con un mensaje que describe las modificaciones realizadas.
4. **¿Cómo puedes ver el historial de commits en Git?**
   - **Respuesta**:
5. **¿Qué significa hacer un `git push`?**
   - **Respuesta**:

6. **Clona un repositorio desde GitHub y navega al directorio del proyecto.**
   - **Respuesta**:
   - `git clone https://github.com/<usuario>/<nombre_repositorio>.git`
   - `cd <nombre_repositorio>`
7. **Crea una nueva rama llamada `feature/agregar-login` y cámbiate a ella.**
   - **Respuesta**:
   - `git checkout -b feature/agregar-login`
8. **Realiza cambios en el archivo `index.html`, añade el archivo al área de preparación y haz un commit.**
   - **Respuesta**:
   - `git add index.html`
   - `git commit -m "Agregado formulario de login"`
9. **Sube los cambios de la rama `feature/agregar-login` al repositorio remoto.**
   - **Respuesta**:
   - `git push origin feature/agregar-login`
10. **Crea una Pull Request en GitHub para fusionar la rama `feature/agregar-login` a la rama `main`.**
    - **Respuesta**: (Pasos dentro de GitHub)

    1. Dirígete al repositorio en GitHub.
    2. Haz clic en "Pull Requests" y luego "New Pull Request".
    3. Elige las ramas correspondientes y crea la PR.
11. **Fusión de la Pull Request realizada y confirma la fusión en GitHub.**
    - **Respuesta**: (Pasos dentro de GitHub)
        1. En la PR, haz clic en "Merge Pull Request" y luego en "Confirm Merge".
12. **Elimina la rama `feature/agregar-login` después de fusionarla.**
    - **Respuesta**:
    - `git branch -d feature/agregar-login`
13. **Actualiza tu repositorio local con los últimos cambios de la rama `main`.**
    - **Respuesta**:
    - `git pull origin main`
14. **Resuelve un conflicto de fusión en el archivo `app.js` que se encuentra en conflicto.**
    - **Respuesta**: (Pasos dentro de Git)
        1. Abre el archivo `app.js` y resuelve el conflicto manualmente.
        2. Marca el archivo como resuelto:
        3. `git add app.js`
        4. `git commit -m "Conflicto resuelto en app.js"`
15. **Crea un nuevo archivo `nuevo-archivo.txt`, añádelo y realiza un commit.**
    - **Respuesta**:
    - `echo "Contenido del nuevo archivo" > nuevo-archivo.txt`
    - `git add nuevo-archivo.txt`
    - `git commit -m "Añadido nuevo archivo de texto"`