HERRAMIENTA DE DESARROLLO

Laboratorio: Instalación, Configuración y Comandos Básicos de Git (Control de Versiones)

Objetivo:

El objetivo de este laboratorio es proporcionar una introducción práctica y teórica sobre la **instalación y configuración de Git**, así como los **comandos básicos** para gestionar proyectos de software. Los estudiantes aprenderán a instalar Git, configurarlo y utilizar los comandos más comunes en el control de versiones.

Procedimientos de Instalación y Configuración:

1. Instalación de Git:

- **Windows**:
  - Descargue el instalador desde el sitio oficial: https://git-scm.com/download/win.
  - Ejecute el instalador y siga las instrucciones predeterminadas.
- **macOS**:
  - En la terminal, ejecute:
  - `brew install git`
  - Si no tiene Homebrew, puede descargar Git desde el sitio oficial: https://git-scm.com/download/mac.
- **Linux (Debian/Ubuntu)**:
  - Abra la terminal y ejecute:
  - `sudo apt update`
  - `sudo apt install git`

2. Configuración Inicial de Git:

Después de la instalación, configure Git con tu nombre y correo electrónico (estos serán usados en los commits).

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

Para verificar la configuración:

```
git config --list
```

3. Verificación de la Instalación:

Para verificar que Git se ha instalado correctamente, ejecute el siguiente comando en la terminal:

```
git --version
```

Esto debería mostrar la versión instalada de Git.

Comandos Básicos de Git:

1. Inicializar un Repositorio:

Para crear un repositorio en el directorio actual:

```
git init
```

2. Clonar un Repositorio Remoto:

Para clonar un repositorio remoto a tu máquina local:

```
git clone <URL_del_repositorio>
```

3. Ver el Estado del Repositorio:

Para ver los archivos modificados, añadidos o eliminados:

```
git status
```

4. Añadir Archivos al Staging:

Para añadir un archivo específico:

```
git add <nombre_archivo>
```

Para añadir todos los archivos modificados:

```
git add .
```

5. Realizar un Commit:

Para realizar un commit de los archivos añadidos al Staging:

```
git commit -m "Mensaje descriptivo del cambio"
```

6. Ver el Historial de Commits:

Para ver el historial de cambios:

```
git log
```

7. Crear y Cambiar de Rama:

Para crear una nueva rama:

```
git branch <nombre_rama>
```

Para cambiar a una rama existente:

```
git checkout <nombre_rama>
```

8. Fusionar una Rama:

Para fusionar una rama en la rama actual:

```
git merge <nombre_rama>
```

9. Subir Cambios al Repositorio Remoto:

Para subir tus cambios al repositorio remoto:

```
git push origin <nombre_rama>
```

10. Obtener Cambios del Repositorio Remoto:

Para obtener los últimos cambios del repositorio remoto:

```
git pull origin <nombre_rama>
```

Preguntas

1. **¿Qué es Git y para qué se utiliza?**
   - **Respuesta**: Git es un sistema de control de versiones distribuido utilizado para gestionar los cambios en el código fuente de un proyecto. Permite realizar un seguimiento de los cambios, colaborar con otros desarrolladores y restaurar versiones anteriores del proyecto.
2. **¿Cuál es la diferencia entre `git pull` y `git push`?**
   - **Respuesta**: `git pull` descarga los cambios desde el repositorio remoto a tu repositorio local, mientras que `git push` sube los cambios locales al repositorio remoto.
3. **¿Qué significa "hacer un commit" en Git?**
   - **Respuesta**: .
4. **¿Cuál es la función de las ramas en Git?**
   - **Respuesta**:
5. **¿Cuál es la diferencia entre un repositorio local y un repositorio remoto?**
   - **Respuesta**:
6. **Realiza la configuración de Git con tu nombre y correo electrónico**
   - **Respuesta**:
7. **Crea un repositorio local en tu máquina.**
   - **Respuesta**:
8. **Crea una nueva rama llamada `cambios/login`**
   - **Respuesta**:
9. **Cambia a la rama `cambios/login`**
   - **Respuesta**:
10. **Visualiza el historial de commits**
    - **Respuesta**: