GUÍA DE LABORATORIO N.° 1: SISTEMA DE CONTROL DE VERSIONES

1. Competencias

Al finalizar la aplicación de esta guía, los estudiantes serán capaces de:

- Comprender los conceptos básicos de un sistema de control de versiones (SCV).
- Instalar y configurar Git en un sistema operativo Windows.
- Realizar operaciones básicas de Git, como iniciar un repositorio, hacer commits y crear ramas
- Trabajar de manera colaborativa utilizando Git.
- Usar Git para gestionar versiones en proyectos de desarrollo de software.

2. Fundamento Teórico

2.1 Sistema de Control de Versiones (SCV)

Un **sistema de control de versiones (SCV)** es una herramienta que permite gestionar los cambios realizados en un conjunto de archivos a lo largo del tiempo. Los SCV son esenciales en proyectos de desarrollo de software porque:

- Permiten mantener un historial detallado de los cambios realizados en el código fuente.
- Facilitan la colaboración entre múltiples desarrolladores sin sobrescribir el trabajo de otros.
- Ayudan a revertir a versiones anteriores del proyecto en caso de errores o problemas.

2.2 Git: Introducción

Git es un sistema de control de versiones distribuido que permite a los desarrolladores realizar un seguimiento de los cambios en su código fuente de manera eficiente. Es ampliamente utilizado en proyectos de desarrollo debido a su rapidez y robustez.

- **Distribuido:** Cada desarrollador tiene una copia completa del repositorio, lo que les permite trabajar de manera autónoma y sincronizar sus cambios cuando lo deseen.
- **Rápido y eficiente:** Git maneja grandes volúmenes de datos de manera eficiente, lo que lo hace ideal para proyectos de software grandes.

2.3 Comandos Básicos de Git

A continuación, se detallan los comandos más comunes que se utilizan en Git:

- `git init`: Inicializa un repositorio vacío de Git en el directorio actual.
- `git clone <url>`: Clona un repositorio remoto a tu máquina local.
- `git status`: Muestra el estado del repositorio y los cambios no confirmados.

- `git add <archivo>` : Agrega un archivo o grupo de archivos al área de preparación (staging area).
- `git commit -m "<mensaje>"` : Realiza un commit con los cambios agregados al área de preparación.
- `git push` : Sube los cambios locales al repositorio remoto.
- `git pull` : Trae los cambios del repositorio remoto a tu copia local.
- `git branch <nombre_de_rama>` : Crea una nueva rama.
- `git checkout <nombre_de_rama>` : Cambia a la rama indicada.
- `git merge <nombre_de_rama>` : Fusiona los cambios de una rama con la rama actual.
- `git log` : Muestra el historial de commits.

2.5 Trabajar con Remotos

Para poder colaborar en cualquier proyecto Git, necesitas saber cómo gestionar repositorios remotos. Los repositorios remotos son versiones de tu proyecto que están hospedadas en Internet o en cualquier otra red. Puedes tener varios de ellos, y en cada uno tendrás generalmente permisos de solo lectura o de lectura y escritura. Colaborar con otras personas implica gestionar estos repositorios remotos enviando y trayendo datos de ellos cada vez que necesites compartir tu trabajo. Gestionar repositorios remotos incluye saber cómo añadir un repositorio remoto, eliminar los remotos que ya no son válidos, gestionar varias ramas remotas, definir si deben rastrearse o no y más.

- **Ver Tus Remotos:** Para ver los remotos que tienes configurados, debes ejecutar el comando `git remote`. Mostrará los nombres de cada uno de los remotos que tienes especificados. Si has clonado tu repositorio, deberías ver al menos *origin* (origen, en inglés) - este es el nombre que por defecto Git le da al servidor del que has clonado

2.4 GitHub

GitHub es una plataforma basada en la web que proporciona alojamiento para proyectos que utilizan Git. Además de ser un repositorio remoto para el código fuente, GitHub permite a los desarrolladores colaborar, gestionar proyectos, y revisar el código de otros mediante "pull requests".

3. Aplicación de la Guía

Ejercicio 1: Instalación y Configuración de Git en Windows

1. **Descarga e Instalación de Git**:
    - Ve al sitio web oficial de Git: https://git-scm.com/download/win
    - Descarga el instalador para Windows y ejecútalo.
    - Durante la instalación, mantén las opciones predeterminadas, pero asegúrate de seleccionar la opción que permita ejecutar Git desde la línea de comandos.
2. **Verificación de la instalación**:
    - Abre el símbolo del sistema o PowerShell en Windows.

- Escribe el siguiente comando para verificar la instalación:

```
git --version
```

- Esto debe devolver la versión de Git instalada, lo que confirma que la instalación fue exitosa.

Ejercicio 2: Creación de un Repositorio Local con Git

1. **Inicia un nuevo repositorio local**:
   - Crea una nueva carpeta en tu computadora para tu proyecto.
   - Abre la terminal en esa carpeta y ejecuta el siguiente comando para inicializar un repositorio Git:

     `git init`
2. **Añadir y realizar un commit**:
   - Crea un archivo llamado `README.md` dentro de tu proyecto y agrégale algo de contenido.
   - Usa el siguiente comando para agregar el archivo al área de preparación:

     `git add README.md`
   - Realiza un commit con un mensaje:

```
git commit -m "Primer commit - Agregado README.md"
```

3. **Verifica el estado del repositorio**:
   - Usa `git status` para ver los cambios realizados.

Ejercicio 3: Trabajo con Repositorios Remotos (GitHub)

1. **Crear un repositorio en GitHub**:
   - Inicia sesión en GitHub.
   - Crea un nuevo repositorio público o privado.
2. **Conectar el repositorio local con GitHub**:
   - Configura tu nombre y correo electrónico globalmente (es importante para los commits):
     1. Abre **Git Bash** o **símbolo del sistema**.

2. Ejecuta los siguientes comandos para configurar tu nombre y correo electrónico (estos datos aparecerán como autor de los commits):

```
git config --global user.name "Tu Nombre"
git config --global user.email "tu.email@dominio.com"
```

- En la terminal de tu repositorio local, agrega el repositorio remoto:

`git remote add origin https://github.com/usuario/repositorio.git`

  - **HTTPS**: `https://github.com/usuario/repositorio.git`

- **Obtener la URL del repositorio:** Ve a tu repositorio en GitHub y copia la URL de clonación. Tienes dos opciones:

  - **SSH**: `git@github.com:usuario/repositorio.git`

3. **Sube tus cambios a GitHub**:

- Sube tus commits locales al repositorio remoto en GitHub:

`git push -u origin master`

> El comando `git push -u` se utiliza en Git para enviar los cambios locales a un repositorio remoto, pero con una característica adicional: **establecer un seguimiento** (tracking) entre la rama local y la remota.
>
> `-u` **(o** `--set-upstream` **)**: Esta opción le indica a Git que **asocie** (haga el seguimiento) entre la rama local y la rama remota en el repositorio remoto. Al usar `-u`, Git recordará qué rama remota debe asociarse con la rama local. Esto permite usar comandos más simples en el futuro, como `git push` o `git pull`, sin necesidad de especificar la rama remota cada vez.

Ejercicio 4: Sincronización y Actualización de un Repositorio con `git pull` y `git push`

1. Navega al directorio del repositorio local

2. **Crear una Nueva Rama para Desarrollar:** Crea una nueva rama para trabajar en una funcionalidad o corrección de un error. Llámala `feature/cambio-texto`.

`git checkout -b feature/cambio-texto`

> **Información**
>
> En la siguiente guía se verá con mayor profundidad del tema de ramificación.

3. **Hacer Cambios en la Rama Local:** Abre un archivo en tu proyecto (por ejemplo, `index.html`) y realiza algunos cambios. Por ejemplo, modifica el contenido de un párrafo o agrega un nuevo título.

4. **Confirmar los Cambios:** Una vez que hayas realizado los cambios, guarda el archivo y agrégalo al área de preparación (staging area).

   `git add index.html`

   Luego, realiza un commit de los cambios:

   ```
   git commit -m "Modificar texto en index.html"
   ```

5. **Obtener Cambios del Repositorio Remoto (git pull):** Antes de subir tus cambios, asegúrate de que tu repositorio local esté actualizado con los cambios que otros colaboradores puedan haber realizado. Para hacer esto, usa `git pull`:

   `git pull origin main`

   Este comando trae los últimos cambios de la rama `main` del repositorio remoto y los fusiona con tu rama local. Si hay conflictos, Git te lo informará, y tendrás que resolverlos antes de continuar.

6. **Subir los Cambios al Repositorio Remoto (git push):** Una vez que hayas confirmado tus cambios y resuelto cualquier conflicto (si lo hubo), es hora de enviar tus cambios al repositorio remoto. Usa el siguiente comando:

   `git push -u origin feature/cambio-texto`

   El `-u` en este comando asocia la rama local `feature/cambio-texto` con la rama remota del mismo nombre, lo que te permitirá usar solo `git push` en el futuro.

7. **Verificar el Estado del Repositorio Remoto:** Después de realizar el push, ve a tu repositorio en GitHub (o en la plataforma que estés utilizando) y verifica que tu nueva rama (`feature/cambio-texto`) haya sido subida correctamente.

> **Nota**
>
> - **¿Qué sucede si alguien más ha subido cambios a la rama `main` mientras trabajabas en tu rama?**
>   Esto podría causar conflictos al usar `git pull`. Al realizar el `git pull` primero, Git intentará fusionar esos cambios con tu rama local, y si no es posible hacerlo automáticamente, tendrás que resolver los conflictos manualmente. Este comando (`git pull`), en realidad, es una combinación de dos operaciones:
>
>   1. `git fetch` : Descarga **todos los cambios del repositorio remoto** que aún no han sido traídos a tu repositorio local, incluyendo nuevas ramas, etiquetas, y cualquier actualización en las ramas existentes. No realiza ningún cambio en tu rama local.
>   2. `git merge` : Fusiona los cambios remotos (por ejemplo, de `origin/main`) con tu rama local actual.
>
>   Es decir, después de realizar el `fetch` , `git pull` automáticamente hace un **merge** de los cambios descargados en tu rama actual.
>
> - **¿Por qué es importante hacer un `git pull` antes de hacer un `git push` ?**
>   Para asegurarte de que tu repositorio local esté actualizado con los cambios más recientes del repositorio remoto. Si no lo haces, podrías terminar empujando cambios conflictivos o desactualizados.

4. Actividades Propuestas

1. **Actividad Individual - Configuración de un Repositorio en GitHub**:
   - Crea un repositorio en GitHub y clónalo en tu máquina local. Realiza al menos tres commits con cambios en diferentes archivos y sube esos cambios al repositorio remoto.
2. **Actividad Grupal - Gestión de Rama y Fusión**:
   - En equipo, crean un proyecto en GitHub. Cada miembro debe crear una rama para trabajar en una característica diferente.
3. **Actividad Individual - Revisión de Historial de Cambios**:
   - Haz varios commits en un proyecto y utiliza el comando `git log` para revisar el historial de los cambios. Extrae información sobre los autores, fechas y mensajes de commit. 4- **Actividad Grupal - Sincronización repositorio**
   - Trabaja en equipo con otros compañeros y realicen cambios sobre el mismo archivo del repositorio.
   - Sin usar ramas, sincroniza tu repositorio local con el repositorio remoto utilizando `git pull` antes de subir tus cambios, y luego sube tus propios cambios utilizando `git push` .
   - Resuelve cualquier conflicto de fusión (merge conflict) si ocurre durante el proceso.

5- Investiga sobre como poder conectarse a un repositorio remoto utilizando una clave SSH. Conectate a tu repositorio remoto utilizando una clave SSH.

5. Referencias

- Git - Acerca del Control de Versiones. (n.d.). Retrieved from https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones