**PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL FORMATO GUÍA DE APRENDIZAJE**

1. **IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE** 
   * **Denominación del Programa de Formación**: Técnico en programación de software
   * **Código del Programa de Formación: 3233198**
   * **Nombre del Proyecto:** Construcción de software integrador de tecnologías

orientadas a servicios

* + **Fase del Proyecto:** Ejecución
  + **Actividad de Proyecto:** Introducción practica GIT y CLI
  + **Competencia:** DESARROLLAR LA SOLUCIÓN DE SOFTWARE DE ACUERDO

CON EL DISEÑO Y METODOLOGÍAS DE DESARROLLO

* + **Resultados de Aprendizaje Alcanzar:** CREAR COMPONENTES FRONT-END DEL SOFTWARE DE ACUERDO CON LAS NECESIDADES DEL CLIENTE.
  + **Duración de la Guía:** 4 Horas

1. **PRESENTACIÓN**

**Aprendiz SENA:**

Bienvenido a esta guía de trabajo sobre Git y la terminal de comandos. Este documento está diseñado para personas sin conocimientos previos en la terminal o en Git, con el objetivo de que comprendan las principales bondades de Git y adquieran habilidades básicas para gestionar proyectos de software de manera eficiente.

Para trabajar con Git, es fundamental tener conocimientos básicos en el manejo de la terminal de comandos. Por ello, esta guía incluye una gran cantidad de ejercicios prácticos que ayudarán a los principiantes a familiarizarse con estos conceptos de forma progresiva.

Git es un sistema de control de versiones distribuido que permite rastrear cambios en el código fuente de un proyecto. Su uso es esencial en el desarrollo de software moderno, ya que facilita el trabajo colaborativo, la gestión de versiones y la recuperación de cambios.

La terminal de comandos es una herramienta poderosa que permite interactuar con el sistema operativo mediante líneas de comando. Es ampliamente utilizada para navegar por directorios, ejecutar programas, manipular archivos y automatizar tareas complejas.

Esta guía proporcionará una introducción práctica a ambas herramientas, con actividades estructuradas y una gran cantidad de ejercicios para reforzar el aprendizaje progresivo.

1. **FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE**

Las actividades de esta guía se dividen en varias fases para garantizar un aprendizaje integral:

**A. Actividades de reflexión inicial.**

* + Reflexiona sobre la importancia de llevar un control de versiones en proyectos de desarrollo.
  + Investiga y responde: ¿Cómo podría beneficiar Git a un equipo de desarrollo?
  + Observa el historial de cambios en documentos colaborativos como Google Docs y compáralo con Git.
  + Busca información sobre el uso de sistemas de control de versiones en la industria.
  + Enumera los problemas que podrían surgir al desarrollar un proyecto sin control de versiones.
  + Investiga qué herramientas similares a Git existen y compara sus características.
  + Explica en tus propias palabras la diferencia entre Git y GitHub.

**B. Actividades de contextualización e identificación de conocimientos necesarios para el aprendizaje.**

**Uso Básico de la Terminal**

Ejecuta los siguientes comandos en la terminal para familiarizarte con su funcionamiento:

* pwd – Muestra el directorio actual.



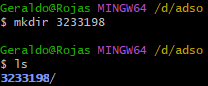
* + ls – Lista los archivos en el directorio actual.



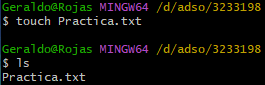
* + cd nombre\_carpeta – Navega entre directorios.



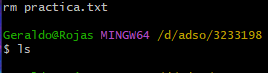
* + mkdir nueva\_carpeta – Crea una nueva carpeta.



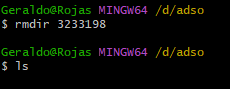
* + touch nuevo\_archivo.txt – Crea un nuevo archivo vacío.



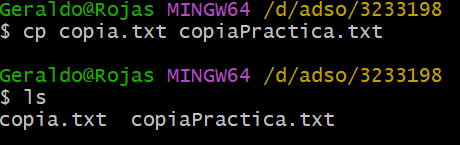
* rm nuevo\_archivo.txt – Elimina un archivo.



* rmdir nueva\_carpeta – Elimina una carpeta vacía.



* + cp archivo.txt copia.txt – Copia un archivo.



* + mv archivo.txt nueva\_ubicación/ – Mueve un archivo.



* + echo "Hola mundo" > mensaje.txt – Escribe texto en un archivo.



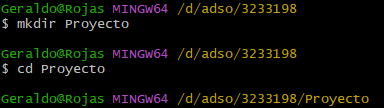
* + cat mensaje.txt – Muestra el contenido de un archivo.



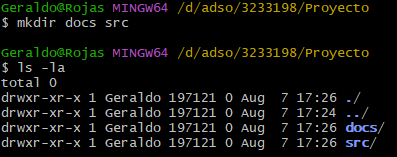
**Ejercicios Prácticos**

Genere los siguientes ejercicios y por cada uno cree una captura de pantalla para llevar el registro del proceso.

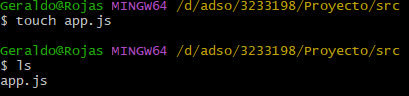
* + Crea una carpeta llamada proyecto y accede a ella.



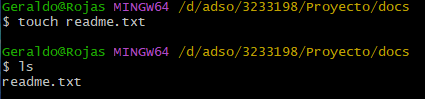
* + Dentro de proyecto, crea dos subcarpetas: src y docs.



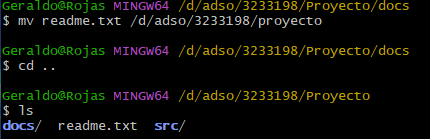
* + Dentro de src, crea un archivo llamado app.js.



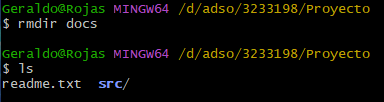
* + Dentro de docs, crea un archivo llamado readme.txt.



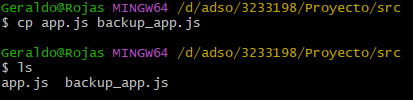
* + Mueve readme.txt al directorio principal proyecto.



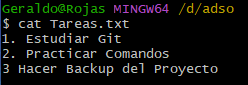
* + Borra la carpeta docs.



* + Copia app.js dentro de proyecto con otro nombre (backup\_app.js).



* + Crea un archivo de texto con una lista de tareas y luego visualízalo con cat. 



* + Usa grep para buscar una palabra en un archivo de texto.



* + Usa find para localizar un archivo dentro de una carpeta.



**C. Actividades de apropiación del conocimiento.**

Ejecuta los siguientes comandos para comenzar a usar Git y por cada comando genere una captura de pantalla.

* + git --version – Verifica si Git está instalado.



* + git config --global user.name "Tu Nombre" – Configura tu nombre de usuario.



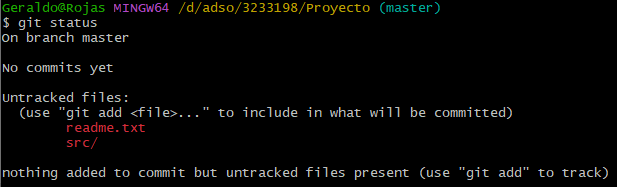
* + git config --global user.email "tuemail@example.com" – Configura tu correo.



* + git init – Crea un nuevo repositorio Git en una carpeta.



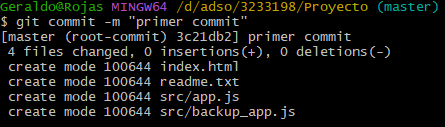
* + git status – Muestra el estado actual del repositorio.



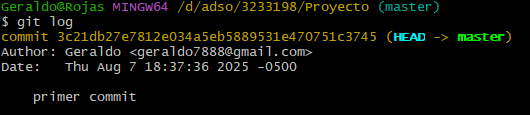
* + git add archivo.txt – Añade un archivo a la zona de preparación.



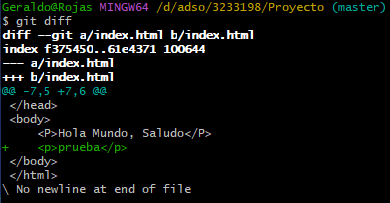
* + git commit -m "Primer commit" – Guarda los cambios en el historial.



* git log – Muestra el historial de commits.

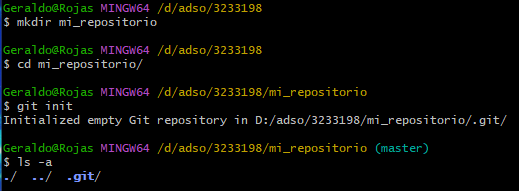


* git diff – Muestra las diferencias entre versiones de un archivo.

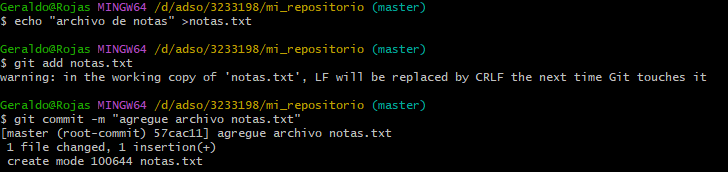


**Ejercicios Prácticos**

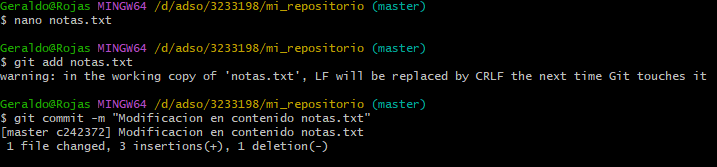
* + Crea un directorio llamado mi\_repositorio y conviértelo en un repositorio Git.



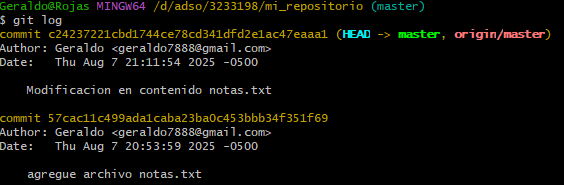
* + Dentro de mi\_repositorio, crea un archivo notas.txt y agrégalo al control de versiones.



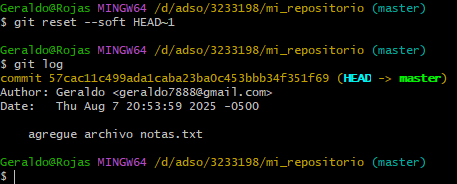
* + Modifica notas.txt, vuelve a guardarlo y realiza otro commit.



* + Usa git log para ver el historial de cambios.



* + Deshaz el último commit sin borrar los cambios con git reset --soft HEAD~1.



* + Usa git checkout -- notas.txt para descartar cambios no confirmados en un archivo.



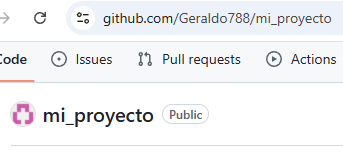
**D. Actividades de transferencia del conocimiento.**

**Trabajo con Repositorios Remotos**

* + Crea una cuenta en GitHub si no tienes una.



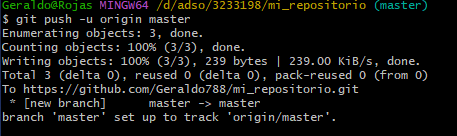
* + Crea un repositorio en GitHub llamado mi\_proyecto.



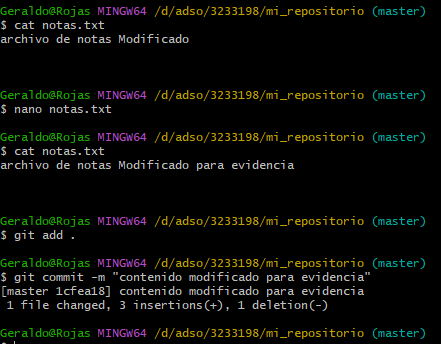
* + Usa git remote add origin <URL> para enlazarlo con tu repositorio local.



* + Usa git push -u origin main para subir tus cambios a GitHub.

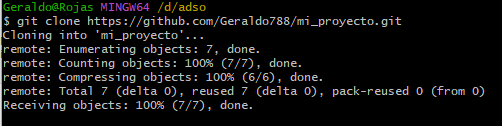


* + Realiza una modificación en notas.txt, haz commit y súbelo nuevamente

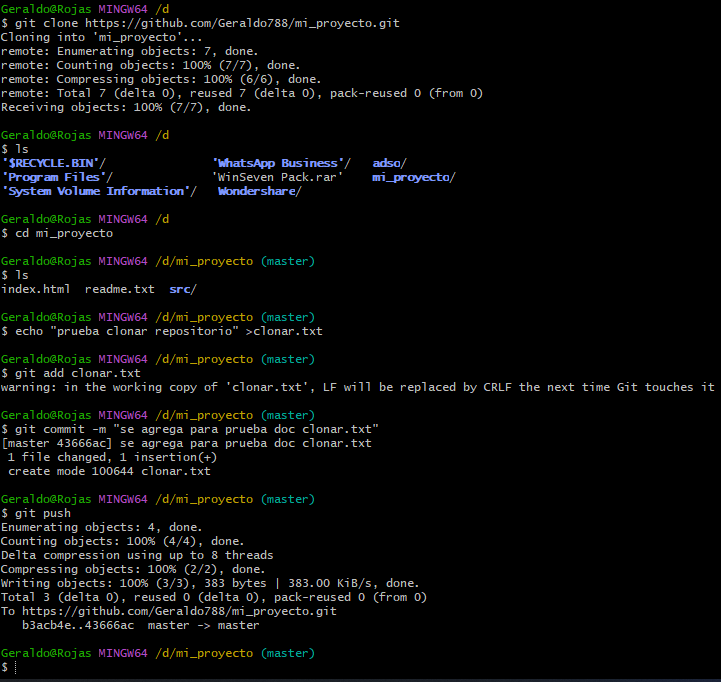


**Ejercicios Prácticos**

* + Clona un repositorio de GitHub en tu computadora.



* + Realiza cambios en el repositorio clonado y súbelos con git push.

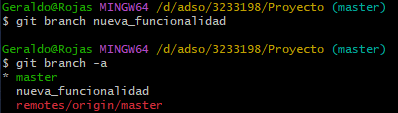


* + Explora la función git pull para actualizar tu repositorio local.

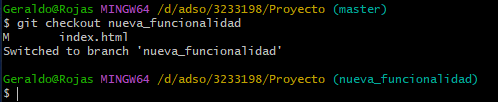


**Manejo de Ramas**

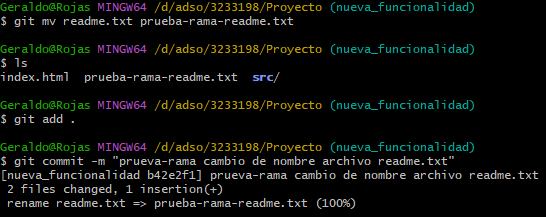
* + Crea una nueva rama llamada nueva\_funcionalidad.



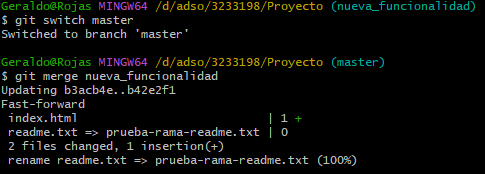
* + Cambia a la nueva rama con git checkout nueva\_funcionalidad.



* + Realiza cambios en un archivo y haz un commit.



* + Cambia de nuevo a la rama main y fusiona nueva\_funcionalidad con git merge.



**Ejercicios Adicionales**

Simula un conflicto de fusión editando la misma línea en main y en nueva\_funcionalidad.

Resuelve el conflicto manualmente y haz commit de los cambios corregidos.

* + Elimina la rama nueva\_funcionalidad con git branch -d nueva\_funcionalidad.

1. **ACTIVIDADES DE EVALUACIÓN**

|  |  |  |
| --- | --- | --- |
| **Evidencias de Aprendizaje** | **Criterios de Evaluación** | **Técnicas e Instrumentos de**  **Evaluación** |
| **Evidencias de**  **Conocimiento:**  **Evidencias de Desempeño:**   * Evaluación(es). * Taller(es).   **Evidencias de Producto:**   * Código fuente * Repositorio remoto y local. * Link de publicación |  | **Técnica:** Lista de chequeo |

1. **GLOSARIO DE TÉRMINOS**

* + **GIT:** sistema de control de versiones distribuido ampliamente utilizado para el seguimiento de cambios en el código fuente durante el desarrollo de software.
  + **Compilación:** proceso de traducir el código fuente de un programa informático a un formato ejecutable que la computadora puede entender y ejecutar.
  + **Repositorio:** Un repositorio en el contexto de la informática, especialmente en el desarrollo de software, se refiere a un lugar centralizado donde se almacenan y gestionan archivos y recursos relacionados con un proyecto específico.
  + **Staging área:** El área de staging, también conocida como "área de preparación" o

"índice", es una etapa intermedia en el proceso de trabajar con Git. Cuando realizas

cambios en tus archivos de código fuente y estás listo para confirmar esos cambios en Git, primero los "añades" al área de staging antes de "confirmar" definitivamente esos cambios en el historial de Git.

* + **Commit:** En Git, un "commit" es una operación que registra los cambios realizados en los archivos de un repositorio en un punto específico de tiempo. Cada commit tiene un mensaje descriptivo que proporciona información sobre los cambios realizados en ese commit en particular. Los commits son la unidad básica de trabajo en Git y forman parte del historial de versiones del repositorio.
  + **Branch:** En Git, una "branch" (rama) es una línea independiente de desarrollo que permite a los desarrolladores trabajar en características específicas del proyecto sin afectar el flujo de trabajo principal o "master". Cada rama en un repositorio de Git representa una versión separada del código, y los cambios realizados en una rama no afectan a otras ramas hasta que se fusionan.
  + **Merge:** En Git, el comando merge se utiliza para combinar los cambios de una rama (branch) en otra. Esta operación se realiza típicamente cuando se quiere incorporar el trabajo realizado en una rama secundaria de vuelta a la rama principal (por lo general, master o main).
  + **Deploy:** El término "deploy" (despliegue) en el contexto del desarrollo de software se refiere al proceso de implementar y poner en funcionamiento una aplicación o una actualización de software en un entorno de producción, donde los usuarios finales pueden interactuar con ella. Este proceso implica tomar el código fuente de la aplicación, compilarlo (si es necesario), configurar cualquier infraestructura requerida, y finalmente lanzar la aplicación para su uso público.

1. **REFERENTES BILBIOGRÁFICOS**

[**https://git-scm.com/doc**](https://git-scm.com/doc)

1. **CONTROL DEL DOCUMENTO**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Nombre** |  | **Cargo** |  | **Dependencia** |  | **Fecha** |  |
| **Autor (es)** | | **John Freddy Becerra**  **Castellanos** | | **Instructor** | | **CIMI** |  | **17 – febrero - 2024** | |

1. **CONTROL DE CAMBIOS** (diligenciar únicamente si realiza ajustes a la guía)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Nombre** | **Cargo** | **Dependencia** | **Fecha** | **Razón**  **Cambio** | **del** |
| **Autor**  **(es)** |  |  |  |  |  |  |