Objetivo de la actividad:

Introducir al estudiante en el concepto de control de versiones, permitiéndole el manejo de una herramienta de control de versiones que orquesten una serie de servicios sencillos alrededor del versionamiento.

Descripción de la actividad

En este ejercicio vamos a utilizar git como sistemas de control de versiones , realiza los ejercicio y da respuesta a los interrogantes a lo largo de este taller.

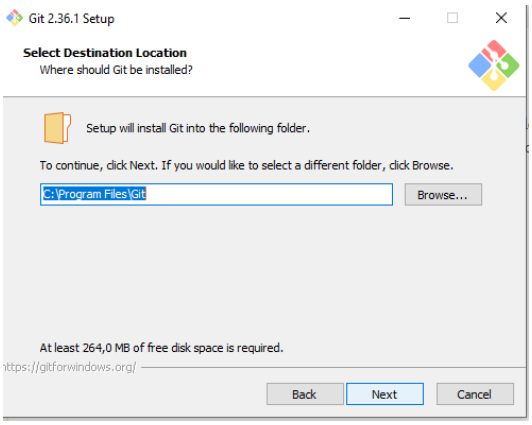
Para iniciar la actividad vamos a realizar algunas consultas:

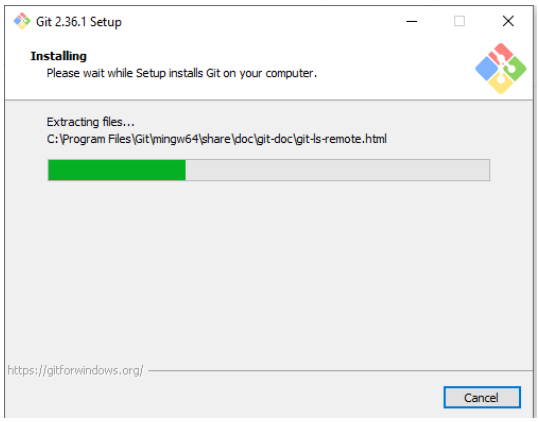
- Cual es la importancia de utilizar un control de versiones.
- Investiga al menos tres plataformas que nos permitan realizar versionamiento de código diferentes a git, de estas investiga sus características y compáralas entre si.
- Que es Git y GitHub?, Cual es la diferencia entre Git y GitHub?, establece sus características.

1. Intalación y configuración de Git.

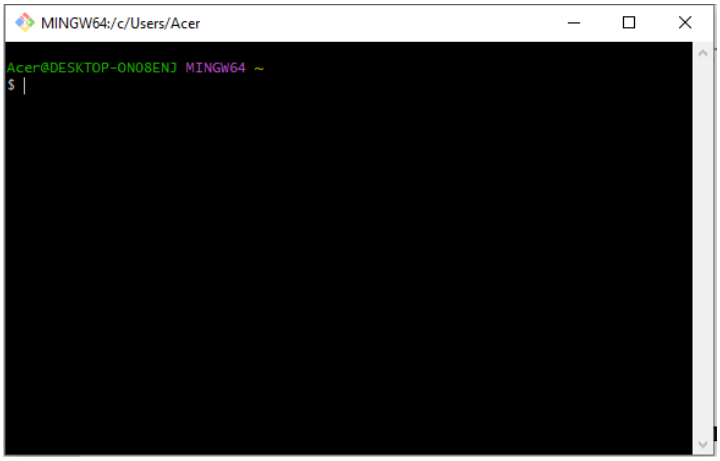
a. Realiza la instalación de Git , utilizando la siguiente página: http://git-scm.com/downloads

Evidencia el paso a paso de la instalación, agrega y describe los pasos.



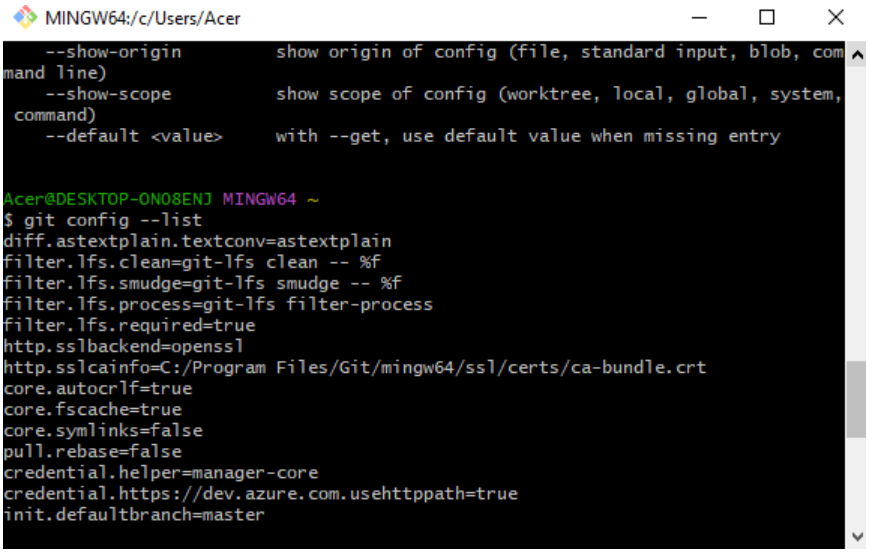


Una vez realizada la instalación, abre tu Git Bash, esta la encuentra en tu grupo de programas Git , en todos los programas



b. En esta consola deberás realizar la configuración de Git, es importante que configures el nombre y el correo electrónico que aparecerá cuando realices los commits que se realizarán sobre los repositorios.
Verifica el estado de Git.

git config –list



git config --global user.name "Nombre"

git config --global user.email correo@electronico.com

En este caso escribe tu nombre y tu correo electrónico institucional, luego evidencia el cambio utilizando git config-list.

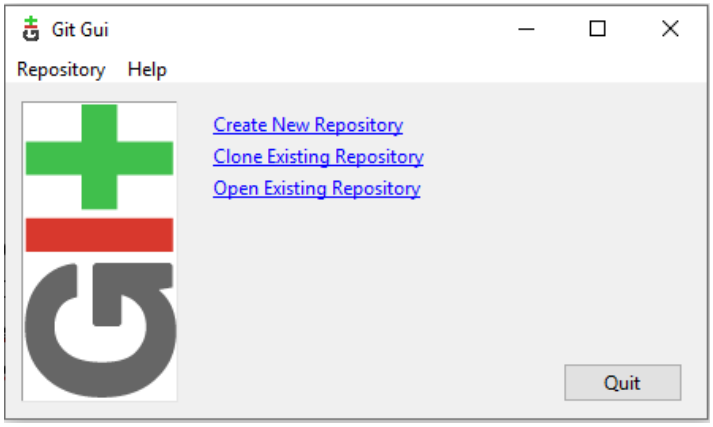
2. En un repositorio Git podemos encontrar varias secciones , investiga y explica las acciones y comandos que utilizamos en cada una de ellas.

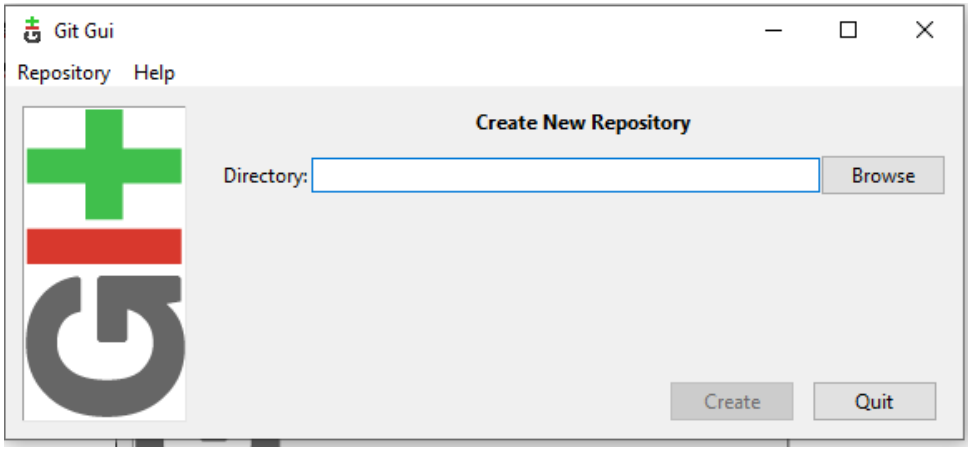
   - Workspace
   - Staging area (Index)
   - Local repository
   - Remote repository

3. Creación de repositorio local
   a. Para crear un repositorio git se utiliza el siguiente comando.

   Git init

   ó puedes utilizar la interfaz grafica Git Gui





Crea un repositorio con el nombre " Ejercicionombre", ejemplo EjercicioLaudyt, personalizas con tu nombre

Recuerda deberás crear el directorio, ingresar a el e inicializar el repositorio.
mkdir Ejercicionombre
cd Ejercicionombre
git init

```
Acer@DESKTOP-ONO8ENJ MINGW64 ~
$ mkdir EjercicioLaudyt

Acer@DESKTOP-ONO8ENJ MINGW64 ~
$ cd EjercicioLaudyt

Acer@DESKTOP-ONO8ENJ MINGW64 ~/EjercicioLaudyt
$ git init
Initialized empty Git repository in C:/Users/Acer/EjercicioLaudyt/.git/

Acer@DESKTOP-ONO8ENJ MINGW64 ~/EjercicioLaudyt (master)
$
```

Lista el contenido del repositorio
Dir git

b. Otros comandos básicos

Los archivos .git pueden tomar los siguientes estados,

- Sin *seguimiento* _
- Preparado (*escenificado*)
- Modificado (*modificado*)
- Confirmado

Verifiquemos el estado del archivo utlizando

git status

con el comando git add <nombre_archivo>, puedes añadir a la staging area un archivo que no este en seguimiento ni modificado.

Si cuentas con varios archivos que queremos mover a la staging area no es necesario hacerlo uno a uno, podemos usar el comando para moverlos todos a la vez:

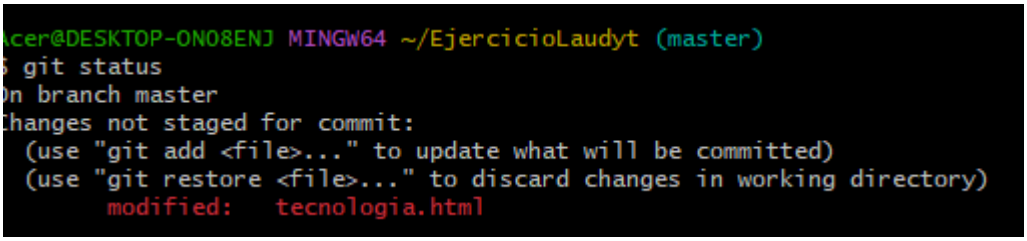
git add -A

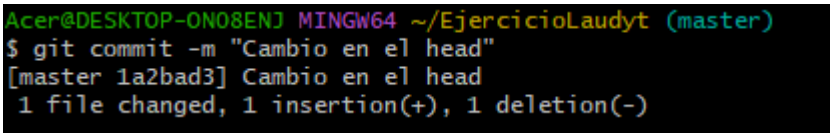
Para mover el archivo de la ***staging area*** al repositorio deberás hacer un ***commit*** .

git commit -m "Escribe un comentario que de cuenta de los cambios realizados"

b.1. Para esta evidencia cree un archivo .html guárdalo en el repositorio, verifique el estado del archivo (git status), añade el archivo a la staging área con (git add<nombre del archivo>), luego realice un commit sobre este, indique un comentario y por último vuelva a realizar revisión del estado del archivo, evidencia con imágenes este proceso

b.2. Ahora realiza un cambio sobre el archivo html en su código, vuelve a revisar el estado del archivo , utiliza los comandos necesarios para evidenciar el cambio realizado, evidencia con imágenes el proceso.





c. Comandos para deshacer cambios

Modifica el texto del último compromiso

git commit -m "Modifico el texto del último commit" –amend

Añade un archivo al último compromiso

git commit –amend

Ejercicio

Crea en tu carpeta un archivo llamado prueba.txt y agrégalo al último commit realizado. Sigue los comandos a continuación.

Git add prueba .txt

git commit -m "Añadimos el prueb.txt"

Para verificar los diferentes commit realizados utiliza

Git log

```
Acer@DESKTOP-ON08ENJ MINGW64 ~/EjercicioLaudyt (master)
$ git log
commit c2e54eab186a907a671da721614d90cb4cc09331 (HEAD -> master)
Author: Laudyt <laudyt.lambrano@cecar.edu.co>
Date:   Fri Jun 3 02:18:22 2022 -0500

    se agregó prueba

commit 3d0f56abe23249a31cb4ce95d29986cad044d2e3
Author: Laudyt <laudyt.lambrano@cecar.edu.co>
Date:   Fri Jun 3 01:56:22 2022 -0500

     Agregando un nuevo archivo para una prueba en el taller

commit 5b807fe20b3c6648a0c673962622fd49ace0905e
Author: Laudyt <laudyt.lambrano@cecar.edu.co>
Date:   Fri Jun 3 00:19:06 2022 -0500

    Cambio en el head

commit ff87af7c1500aa8db3d576055747a3865e58a0ae
Author: Laudyt <laudyt.lambrano@cecar.edu.co>
```

Si queremos crear otra rama en donde

Primero utilizamos git branch y verificamos que solo tenemos una línea master

```
Acer@DESKTOP-ONO8ENJ MINGW64 ~/EjercicioLaudyt (master)
$ git branch
* master
```

Ahora podemos crear una versión alternativa a la cual le vamos a colocar el nombre junior, para lo cual utilizaremos git branch junior, al escribir git Branch podemos identificar las 2 versiones.

```
Acer@DESKTOP-ONO8ENJ MINGW64 ~/EjercicioLaudyt (master)
$ git branch junior

Acer@DESKTOP-ONO8ENJ MINGW64 ~/EjercicioLaudyt (master)
$ git branch
  junior
* master
```

Ahora nos cambiaremos a la versión junior, para ello utilizaremos git checkout junior.

```
Acer@DESKTOP-ONO8ENJ MINGW64 ~/EjercicioLaudyt (master)
$ git checkout junior
Switched to branch 'junior'

Acer@DESKTOP-ONO8ENJ MINGW64 ~/EjercicioLaudyt (junior)
$
```

Una vez ubicado en la versión junior realiza cambios sobre el archivo uno de los archivos y se deberá crear un nuevo archivo llamado Prueba2.txt , deberás guardar los cambios en esta versión.

Utiliza git add <nombre del archivo >, para colocar el archivo en ***staging area*** , luego utiliza commit.

Para evidenciar los cambios en cada versión utiliza git log evidencia con imágenes.

De esta forma podemos crear diferentes versiones del proyecto realizado y llevar el control de los cambios realizados.

```
$ git log
commit 54bd7fd58e9517b9456f7b6f93749e66441ebfcb (HEAD -> junior)
Author: Laudyt <laudyt.lambrano@cecar.edu.co>
Date:   Fri Jun 3 03:30:48 2022 -0500

    cambio en prueba2

commit c2e54eab186a907a671da721614d90cb4cc09331 (master)
Author: Laudyt <laudyt.lambrano@cecar.edu.co>
Date:   Fri Jun 3 02:18:22 2022 -0500

    se agregó prueba

commit 3d0f56abe23249a31cb4ce95d29986cad044d2e3
Author: Laudyt <laudyt.lambrano@cecar.edu.co>
Date:   Fri Jun 3 01:56:22 2022 -0500

    Agregando un nuevo archivo para una prueba en el taller

commit 5b807fe20b3c6648a0c673962622fd49ace0905e
Author: Laudyt <laudyt.lambrano@cecar.edu.co>
Date:   Fri Jun 3 00:19:06 2022 -0500

    Cambio en el head
```

4. Trabajando con repositorios remotos.

Existen dos formas de trabajar con un repositorio remoto.

a. Cuando clonamos el repositorio remoto en nuestra máquina.

git clone <url_del_repositorio_remoto>

Ejemplo.

git clone https://github.com/llambrano1/EjercicioElectiva.git

b. Cuando ya tenemos creado un repositorio local y queremos agregar un repositorio remoto para sincronizarnos.

git remote add <alias> <url_del_repositorio_remoto>

Ejemplo

git remote add EjercicioElectiva https://github.com/llambrano1/EjercicioElectiva.git

Para realizar esta actividad deberás trabajar con repositorios remotos.

b.1. Ingresa a GitHub https://github.com y crea una cuenta a su vez crea un repositorio , evidencia con imágenes este proceso.

b.2. añade el repositorio remoto al repositorio local

```
Acer@DESKTOP-ONO8ENJ MINGW64 ~/EjercicioLaudyt (master)
$ git remote add EjercicioElectiva https://github.com/llambrano1/EjercicioElecti
va.git
```

Para comprobar si el repositorio remoto se ha añadido correctamente ejecuta.

git remote -v

El comando anterior nos devolverá estas dos líneas:

```
Acer@DESKTOP-ONO8ENJ MINGW64 ~/EjercicioLaudyt (master)
$ git remote -v
EjercicioElectiva       https://github.com/llambrano1/EjercicioElectiva.git (fet
ch)
EjercicioElectiva       https://github.com/llambrano1/EjercicioElectiva.git (pus
h)
```

Estas dos líneas finalizan con fetch y push, investiga que significan cada una?

Comando utilizados para trabajar en repositorios remotos.

Ademas de los comados utlizados en repositorios locales , podemos utilizar

Git push

para enviar al repositorio remoto los commits que hemos hecho en nuestro repositorio local. La forma más habitual de usar es hacerlo después de cada commit

```
Acer@DESKTOP-ON08ENJ MINGW64 ~/EjercicioLaudyt (master)
$ git push EjercicioElectiva
info: please complete authentication in your browser...
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 4 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (15/15), 2.32 KiB | 339.00 KiB/s, done.
Total 15 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/llambrano1/EjercicioElectiva/pull/new/master
remote:
To https://github.com/llambrano1/EjercicioElectiva.git
 * [new branch]      master -> master
```

Al realizar este taller has utilizado de forma muy simple las herramientas git y github, en donde has creado diferentes versiones con sus respectivos archivos y sus cambios.

5. Investiga sobre git flow. ¿Que es?, como se utiliza y sus principales comandos.

   5.1 En base a la investigación y teniendo en cuenta el ejercicio principal demuestra el uso de git flow como alternativa de creación de ramas en git.

6. Ejercicio

Realiza un formulario de ingreso a una aplicación, crea un repositorio en Git , define una estrategia de ramas que usarás en tu equipo , crea una rama y haz al menos 3 commits en ella.

Presenta un documento en pdf (máximo 30 hojas) que evidencie el ejercicio realizado y de respuesta a las preguntas realizadas a lo largo de este archivo.

Control de versiones : nota 5.0	**Descripción**	**Puntuación máxima (puntos)**	**Peso %**
Desarrollo de los pasos de la actividad	Desarrollar cada uno de los pasos de la actividad completamente y de manera adecuada.	5	50 %
Calidad de las respuestas	Responder adecuadamente a cada uno de los interrogantes planteados en el enunciado del taller y los puntos prácticos (el taller no deberá tener más de 20% de uso de IA)	3	30 %
Calidad del Reporte	• Buena estructura del documento en términos de títulos. • Trabaja bien la ortografía y gramática • Adopta correctamente las normas APA. • Se ajusta a la longitud del documento.	2	20 %
		10	**100 %**