Curso Git

¿Qué es git?

.Un sistema de control de versiones que registra los cambios realizados en un archivo o conjunto de archivos

.Poder cambiar entre versiones y corregir errores, también con comets ayudas a organizar el proyecto

Tres estados y su Flujo de Trabajo

-Working Directory 1º

- ❏ Cuando estás escribiendo código

-Staging area 2º

- Donde se selecciona los archivos que están listos para el 3 estado, igual que decidimos que archivos no están listos por el momento

-Repository 3º

- ★ En este repositorio se guardará todo el proyecto

Configuraciones Básicas

(git config [<options>])

- -Nombre de user
  git config --global user.name "AgusBaez**"**
- -email de user
  git config --global user.email "agustinbaezignacio@gmail.com**"**

colores para el cmd de git:

- **git config --global color.ui true**

- Limpiar la pantalla del cmd
  clear

Comandos en consola de Git

Los comandos que más se usan en git son:

- "Git status", "git log", "git add" , "git commit -m" , "git push origin master", "git branch", "git merge", "ls", "exit" y "clear".

- **git init** - Marca el inicio de nuestro proyecto, en ese momento git comenzará a guardar nuestro proyecto y sus cambios

- **git status** - nos mostrará el estado de nuestro proyecto, por ejemplo archivos que aún no han sido añadidos a un commit

- Para eso usamos **git add <NOMBRE DEL ARCHIVOS>**

- **git add -A** agregara todos los archivos

- **git commit -m "**Mensaje**"** Se guardarán los cambios realizados en el proyecto con un mensaje para identificarlos
    - ***git log*** Sirve para darnos una lista de los commits generados con su respectiva información
    - ***git log > commits.txt*** Nos añadirá un archivo .txt con todos los comits que hayamos hecho

- ***git checkout <ID DEL COMMIT/RAMA>***con este comando podremos regresar a antiguos commits o ramas
    - Para crear y moverse a una rama deberás utilizar **git checkout -b <Nombre de la rama>**

- ***git reset*** Si usamos este código a diferencia del checkout este eliminará los comits que hayamos creado
    - Si usamos ***git reset --soft*** No tocara nuestra área de trabajo(el código actual)
    - **git reset --mixed** este borrara el área de ensayo
    - **git reset --hard** borra todo
- **git branch**
    - nos mostrará todas nuestras ramas y nos coloreara en la que estemos actualmente más un *****
        - Si queremos crear una rama deberías de escribir en la consola **git branch <NOMBRE DE LA RAMA>**

    - **git branch -D <NOMBRE DE LA RAMA/COMMIT>**

Comandos o Teclas para el movimiento en la Consola

Q - sirve para regresar a la vista anterior
cd.. nos sirve para retroceder tambien cd <Nombre de la carpeta>Para movernos hacia un lugar específico
ls lista de archivos en carpeta

Conceptos

Head

El término **head** refiere al commit en el que nos encontremos

Ramas

Las **ramas** hacen mención a una "línea de tiempo" en nuestro proyecto, estas **ramas** nos servirán para arreglar errores, experimentar, hacer grandes cambio y alguna que otra cosa

- ***Rama Master***

- A esta rama se le llama asi por que es en la cual nosotros comenzaremos a trabajar nuestro proyecto, esta misma será nuestra rama principal y estable de nuestro proyecto

Fusiones

¿Cómo funciona?
Se crea una Fusión cuando cuando se crea un nuevo commit juntando una rama con otra

¿Cómo se hace?
Primero nos situaremos en la rama que queremos integrar los cambios
Por ejemplo nos situamos en la rama con **git checkout <nombre de la rama>**

Una vez situados en la rama que queres que absorba los cambios usaremos
git merge <nombre del commit> te mostrará también los cambios que se hicieron en la rama

Cuando se crea una función existen 2 variantes

-Fast-Forward: Simple y Automático

-Manual Merge: Largo y Mnual

GitHub

¿Qué es GitHub?

- Para aclarar ideas, git es un sistema de control de versiones y GitHub Es una plataforma de guardado de proyectos, usando git podremos gestionar los proyectos del mismo

-git clone <copiar el tipo de URL: HTTPS/SSH> Toma un proyecto de GitHub y nos lo clona a nuestra computadora

Que debemos de hacer para tener nuestro **Repositorio** en GitHub

¿Qué son los **repositorios**?

- Tenemos 2 tipos de repositorios, lo que son Locales y los que se denominan Remotos
    - Los Repo locales son los que se almacenan en nuestra computadora
    - Y los Repo que se almacenan en GitHub son Remotos

Ahora una vez teniendo cuenta en github podremos acceder a +New Repositorio en el cual podremos dar nombre y descripción al proyecto dentro de GitHub

Con **git remote** vinculamos nuestro proyecto local al remoto

- **git remote add origin <copiar el tipo de URL: HTTPS/SSH>**
    - En caso de queramos ver o comprobarlo usamos git remote -v
    - Si queremos eliminarlo usaremos git remote remove origin