**Buenos Día Inge/Compañeros.**

**1 Para qué nos sirve un Sistema de control de versiones Git**

**Git es una herramienta que realiza una función del control de versiones de código de forma distribuida**.**Sus características son**.

Es muy potente

Fue diseñada por Linus Torvalds

No depende de un repositorio central

Es software libre

Con ella podemos mantener un historial completo de versiones.

**Investigar sus funcionalidades y algunos comandos básicos.**

Fundamentalmente, un proyecto Git se estructura en tres partes o tres cajas:

El área del working directory, que es dónde vamos a tener todos nuestros ficheros, dónde estamos trabajando constantemente.

El staging area, que es donde van los archivos que estamos modificando y que aceptamos para que vayan en una futura revisión.

El área de commit o el git directory, que es dónde se almacenan la revisión completa. A lo largo de nuestro curso de Git, se explicará cómo podemos movernos a lo largo de esos tres estados, para qué sirven y por qué suponen una ventaja.

**Comandos Básicos de GIT**

git add [archivo] Agrega el archivo al Stage

git rm --cached [archivo] Quita el archivo del Stage

git add -A Agrega todos los archivos al Stage

git rm -f [archivo] Quita los archivos del Stage y borrarlos

git commit -m "[mensaje]" Realiza commit al archivo y envia un mensaje para saber que se

hizo ejm. ‘Iniciar nuestro landing’

git log Muestra el historial de todos los commit que hemos realizado

git log --oneline Version resumida del historial

git log -[numero] Muestra la cantidad de commits segun el numero que indiques

git commit --amend Anexa al ultimo commit, un cambio que hayamos olvidado hacer.

git tag -a [version] -m "[mensaje]" Crea un tag (etiqueta) a el último commit realizado

git tag [version] [hash1] Crear un tag a un hash especifico

git tag -l Muestra todos los tag

git tag -d [version] Elimina el tag

git branch [nombre] Crea una nueva rama (Master también se considera una rama)

git branch -l Lista todas las ramas que tienes

git branch -d Borra la rama si esta esta vacía.

git branch -D Borra la rama aún asi esta contenga información.

git branch -m [NombreActual][NombreNuevo] Cambia de nombre a una rama

git config --list Muestras todas las propiedades que git a configurado

git diff [hash1] Muestra las diferencias entre el hash que has puesto con el ultimo que

tenemos

git reset --soft Quita un cambio, pero lo mantiene en el Stage.

git reset --mixed Quita un cambio, lo quita del stage. Los mantiene en el working directory

git reset --hard Quita un cambio, los borra totalmente, pero si tienes guardado el hash1,

lo puedes recuperar

git checkout [branch] Nos permite pasearnos entre los branchs (ramas), tambien crearlas.