**GIT Y GITHUB**

Los sistemas de control de versiones son programas que tienen como objetivo controlar los cambios en el desarrollo de cualquier tipo de software, permitiendo conocer el estado actual de un proyecto, los cambios que se le han realizado a cualquiera de sus piezas, las personas que intervinieron en ellos, etc.

Git es el sistema de control de versiones más conocido y usado actualmente y es el motor de Github.

Sirve para:

* Comparar el código de un archivo, de modo que podamos ver las diferencias entre versiones
* Restaurar versiones antiguas
* Fusionar cambios entre distintas versiones

Trabajar con distintas ramas de un proyecto, por ejemplo, la rama de producción (la misma que tiene el cliente instalado), la rama de testeo (la rama que tiene el grupo de testing), la rama de desarrollo (la rama que usan los programadores).

Se puede trabajar mediante dos variantes:

**Sistemas centralizados:**En estos sistemas hay un servidor que mantiene el repositorio y en el que cada programador mantiene en local únicamente aquellos archivos con los que está trabajando en un momento dado y se utiliza cuando se necesita conectar con el servidor donde está el código para poder trabajar y enviar cambios en el software que se está programando. Ese sistema centralizado es el único lugar donde está todo el código del proyecto de manera completa. Subversión o CVS son sistemas de control de versiones centralizados.

**Sistemas distribuidos:**En este tipo de sistemas cada uno de los integrantes del equipo mantiene una copia local del repositorio completo. Al disponer de un repositorio local, puedo hacer commit (enviar cambios al sistema de control de versiones) en local, sin necesidad de estar conectado a Internet o cualquier otra red. En cualquier momento y en cualquier sitio donde esté puedo hacer un commit. Es cierto que es local de momento, luego podrás compartirlo con otras personas, pero el hecho de tener un repositorio completo me facilita ser autónomo y poder trabajar en cualquier situación**. Git es un sistema de control de versiones distribuido.**

 Los sistemas centralizados permiten definir un número de versión en cada una de las etapas de un proyecto, mientras que en los distribuidos cada repositorio local podría tener diferentes números de versión. También en los centralizados existe un mayor control del desarrollo por parte del equipo.

*En términos comparativos nos podemos quedar con la mayor ventaja de los sistemas distribuidos frente a los sistemas centralizados: La posibilidad de trabajar en cualquier momento y lugar, gracias a que siempre se mandan cambios al sistema de versionado en local, permite la autonomía en el desarrollo de cada uno de los componentes del equipo y la posibilidad de continuar trabajando, aunque el servidor central de versiones del software se haya caído.*

Git tiene tres estados principales en los que se pueden encontrar tus archivos: confirmado (committed), modificado (modified), y preparado (staged).

**Confirmado:** significa que los datos están almacenados de manera segura en tu base de datos local.

**Modificado:** significa que has modificado el archivo pero todavía no lo has confirmado a tu base de datos.

**Preparado:** significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

Esto nos lleva a las tres secciones principales de un proyecto de Git:

El directorio de Git (Git directory),

El directorio de trabajo (working directory),

El área de preparación (staging area).

**GitHub** es un servicio para hacer hosting de repositorios de software que se administra con Git. Digamos que en GitHub mantienes una copia de tus repositorios en la nube, que además puedes hacer disponible para otros desarrolladores.Por tanto, el uso que le daremos es como repositorio remoto. Pero debe quedar claro que primero debo tener los cambios en el repositorio local y luego podré "empujar" esos cambios al repositorio remoto.

En GitHub puedo tener varios repositorios, pero no es necesario tener todos esos repositorios en mi pc, únicamente aquellos con los que voy a trabajar y para hacerlo debo “clonar” el repositorio web en mi pc.

En el momento que los tenemos en local podremos hacer cambios, almacenarlos en nuestro repositorio local y cuando lo juzguemos oportuno, enviarlo (empujar, push) a tantos servidores o repositorios remotos como queramos y a la rama en la que estemos trabajando.

Lo que describimos de GitHub en líneas generales sirve para entender otros servicios de hosting de repositorios Git como Bitbucket.

Instalación:

1° paso

<https://git-scm.com/>

inmediatamente después de instalar Git, lo primero que deberías hacer es lanzar un par de comandos de configuración.(luego de realizar git init sobre la carpeta que queres tener como repositorio local)

Abriendo la comanda de git bash o la consola de Windows:

git config --global user.name "Tu nombre aquí"

git config --global user.email "[tu\_email\_aquí@example.com](mailto:tu_email_aquí@example.com)"

* Para saber que versión tenemos instalada ponemos:

git –version

* para saber donde estoy parado el comando

pwd

* para saber que tengo en la carpeta donde estoy parado

ls

* para saber en detalle lo que tengo en la carpeta

ls -l

* para saber el peso de los archivos

ls -lh

* para limpiar la pantalla

clear

* para moverme entre carpetas uso el comando

cd

cd nombrecarpeta

cd .. (para regresar)

* para crear carpetas dentro de un directorio

mkdir nombrecarpetanueva

* para crear archivos nuevos

touch nombrearchivoconextension

* para ver que tiene un archivo dentro

cat nombrearchivo

* para ver archivos ocultos

ls -la

* para eliminar archivos

rm nombredearchivo

* para eliminar una carpeta voy al directorio que la contiene

rm -r nombredelacarpeta/

* iniciar git

git init

* luego de loguear el mail y el usuario podemos saber quién esta logueado con

git config --global user.name

git config -- global user.email

* para subir los archivos de la carpeta al staging área

git add nombrearchivo

* para verificar el estado de los archivos

git status

* para grabar en el repositorio local

git commit -m “comentario de grabación”

* para saber las cosas que pasan

git log

* para subir a github

git remote add origin direcciondegit

* para ver si esto se hizo bien

git remote -v

* para subir al repositorio remoto

git push origin nombre de la rama de github , ejemplo

git push origin master

git push origin main

git push origin rama2

Videos importantes:

<https://www.youtube.com/watch?v=ptXiQwE535s>

<https://www.youtube.com/watch?v=SIotISUq-3w&list=PLoCpUTIZIYORkDzYwdunkVf-KIqGjyoot&index=2>

<https://www.youtube.com/watch?v=W-z0iTXEarw>

<https://www.youtube.com/watch?v=KAY1j-MPpyA&t=1s>

<https://www.youtube.com/watch?v=NRaLP7vtoOY>

<https://www.youtube.com/watch?v=9p7Avpcs7Bk>