***GIT***

***Definiciones y conceptos***

**SISTEMAS DE CONTROL DE VERSIONES**

Es una herramienta que sirve para almacenar versiones de un proyecto se puede "viajar" a través de las versiones del proyecto cuando se es necesario.

**GIT:** Git es un sistema de control de versiones y es un software libre creado por el Linux Torvalds.

***¿Para qué sirve?***

Facilita el desarrollo y mantenimiento de aplicaciones tanto de forma individual, pero sobre todo al trabajar en equipo.

***¿Como facilita el desarrollo y el mantenimiento de aplicaciones?***

Guardando un registro de todos los cambios que sufren los archivos de un proyecto a lo largo del tiempo. La verdadera potencia de GIT es que permite tomar por decirlo de alguna manera una captura o foto de cómo se encuentra un proyecto en un momento determinado, la potencia radica en que cuando se está creando una aplicación grande con cientos de archivos GIT hace esa especie de "captura de pantalla" a TODO el proyecto, para así no encontrarse con problemas al momento de volver un archivo a una versión determinada y que todo se rompa. Por otro lado, GIT permite crear algo que se llaman ramas las cuales permiten que se pueda trabajar en equipo. Un proyecto se divide en diferentes tareas, todas estas realizadas por varios programadores, cada programador crea una rama para trabajar de forma individual en este y no afectar la aplicación o el proyecto que ya esta en producción, es decir trabaja de forma independiente en sus tareas y al final todos los cambios que hicieron los desarrolladores en cada rama se unen a la rama principal la cual es el proyecto en sí.

**¿Qué es un repositorio?**

Es un proyecto al cual GIT le está haciendo seguimiento. Un repositorio puede crearse desde cero, es decir se comienza un proyecto desde cero y GIT le empieza a hacer seguimiento desde el comienzo o puede suceder que un proyecto que ya ha sido iniciado y tiene una gran parte desarrollada empiece a usar GIT en ese caso solo se hará seguimiento desde el punto en el cual se empezó a usar GIT.

***Staging Area:***

Es un área temporal donde se almacenan los cambios antes de tomarle la captura.

***Comandos básicos de GIT.***

*1.git init- INICIO UN NUEVO REPOSITORIO:* Este comando se usa una sola vez y es al comenzar un proyecto o cuando queremos que GIT haga un seguimiento a un proyecto. Se puede empezar a hacer seguimiento a proyecto ya iniciado.

GIT crea dos áreas donde almacenara los archivos**. Esta el staging area:** Se almacenarán los archivos de forma temporal, se puede ver el estado en el que están los archivos y el otro es **Repositorio local:** Es aquel en el cual se almacenan aquellas fotos fijas que hace GIT al proyecto en ese momento y son las que se pueden rescatar. GIT puede hacer seguimiento a una cantidad especifica si así lo queremos.

2.git add .- AÑADO TODOS LOS CAMBIOS AL STAGEAREA:

AÑADO TODOS LOS CAMBIOS AL STAGEAREA. Este comando indica a git a que archivos debe hacer seguimiento, con esto git lleva los archivos desde el directorio de trabajo al área de ensayo internamente. Es decir, basicamente git add manda los cambios hechos al statege área.

3.git add nombre del archivo o directorio- Añade los cambios SOLO de ese archivo o directorio específico.

4.git commit -m "nombre del comit":

Lo que hace es tomar la instantánea de cómo está el proyecto hasta ese momento, es importante mencionar que antes de hacer cualquier commit, es fundamental haber mandando todos los cambios al statearea porque si no el commit no funcionara, por lo cual antes de hacer un git commit se debe haber hecho un git add.

Este comando lo que hace es trasladar por fin lo que hay en el área de ensayo al área del repositorio local, es ahi donde git toma la instantánea del archivo, ahí se crea el respaldo y podemos ir a este o estos respaldos en cualquier momento.

5.git status: Da un vistazo de cuales son los cambios.

6.git status-s

Este comando nos da un vistazo de cuales son todos los cambios que hemos hecho y no hemos enviado al stagearea, ya sea el haber creado, elminado o modificado un directorio o un archivo.

7.git clon- CLONO UN REPOSITORIO YA EXISTENTE

8.GIT commit -am "nombre del commit"

Este comando sirve para hacer el add y el commit al mismo tiempo

9.git log --oneline

Lo que hace es mostrarme todos los commits que he hecho en el directorio. Saca como una lista de los commits con su codigo respectivo.

10.git reset --hard Código del commit

Lo que hace este comando lo que hace retornar el proyecto hasta el commit que le pasemos como codigo, pero elimina todos los commits que se hayan hecho despues de ese.

11. git reset --soft Código del commit

La diferencia del anterior es que aunque borra el commit, no borra los cambios, es decir lo que hice antes de hacer el commit lo deja, aunque borra el commit los cambios se conservan, con hard borra tanto el commit como los cambios hechos.

Por lo cual puede ser una opción interesante si se desea solo borrar el commit, pero dejar los cambios como tal.

12. git revert: consultarlo

***RAMAS***

Las ramas son espacios separados donde se puede trabajar, se puede entender como flujos aislados que posteriormente se integran en una sola rama

llamada master. La rama master es como el tronco de un árbol, es decir, es la rama principal donde está el proyecto, es la que usa el cliente y está en producción. Por ende, para no afectar esta, se crean ramas alternas donde se puedan hacer las modificaciones pertinentes, para luego integrar esos cambios hechos en la rama a la master. Ejemplo, se puede crear una rama para el Html, otra para el Css y otra para el Javascript. El proceso de unir las ramas externas a la rama master se llama hacer un MERGE. Git permite hacer cambios en la rama master y en las otras. Una rama puede tener ramas hijas, es decir de una rama principal sale una rama paralela y de esa rama pueden salir 3 ramas más y etc.

***Comandos para las ramas***

1.git branch:

Muestra las ramas que existen en todo el proyecto y también muestra en la rama

que estoy actualmente.

2.git branch "nombre de la rama":

Crea una rama nueva".

3.git checkout "nombre de la rama":

Lo que hace es moverse a la rama a la cual le pase como nombre.

4. git merge

Paso 1: Estar seguro de estar en la rama master

Paso 2: git merge nombre de la rama con la que se quiere hacer el merge

5.git branch -d nombre de la rama:

Elimina la rama que se pase como nombre, pero no se puede estar en esa rama, se debe mover a la master o a otra.

En VSCode

control + enter para hacer commit

GIT PULL: Este comando se utiliza para descargar todos los cambios que hay en el repositorio remoto y tener dichos cambios en el repositorio local.

GIT PUSH