**¿Qué es GIT?**

Git es un sistema de control de versiones de software que permite trabajar en equipo llevando un historial de todos los cambios en el proyecto. Varias personas pueden trabajar sobre el mismo proyecto, aportando conocimiento, comentarios, revisiones y cada modificación al proyecto se va guardando como versiones al proyecto original, con el motivo de tener un historial del avance, para consultas o recordatorios de lo que se hizo día a día.

Git funciona con repositorios, es decir proyectos que van a tener un historial de cambios o versiones, y cada uno de esos cambios registrados en el proyecto o repositorio se llaman commit. Adicional Git maneja ramas, que son los caminos que puede tomar el proyecto, teniendo la rama Master que es la que se publicara como la versión original y ramas secundarias que son donde se van realizando las pruebas, correcciones y mejoras, que al final si funcionan se unifican con la rama master o bien si no son funcionales se eliminan, con el motivo de no alterar la rama master, que es la rama que se lleva a producción.

Como Git trabaja en equipo, cada miembro debe clonar el repositorio original y trabajar sobre la copia exacta para no alternar y comprometer el proyecto principal. La clonación se hace en el equipo local del miembro o desarrollador que se integro al equipo. También existen los Fork, estos no son clones, sino repositorios que se originan o se basan a partir de un proyecto original, pero son completamente diferentes.

El flujo de Git, inicia con la creación de un repositorio o la clonación, se clona si se esta integrando al equipo como un desarrollador nuevo, de lo contrario se crea uno. Para crearlo se utiliza el comando: git init, para clonar se utiliza el comando: git clone. Una vez se está trabajando en el repositorio y se hacen cambios se deben enviar como commits al repositorio original, desde el clon local, esto se debe hacer manual, no es automático, pero no se envían directamente sino a un área de preparación llamada, Stagin Area, para esto se utiliza el comando: git add. Una vez se este seguro que lo que se encuentra en el stagin área esta correcto y listo para enviar al repositorio original, se envían los commits utilizando el comando: git commit.

La herramienta para utilizar Git, y realizar lo anterior se hace desde la línea de comandos o terminal, esto no es de forma gráfica, pero existen herramientas graficas que ayudan a facilitar el trabajo de la línea de comandos, entre estas herramientas esta GitHub, que también es un repositorio en la nube.

**¿Qué es GITHUB?**

Es la plataforma online o en la nube que consolida los repositorios trabajados con GIT. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedan descargar la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

Las principales características de la plataforma es que ofrece las mejores características del servicio de control de versiones sin perder la simplicidad, y es una de las más utilizadas del mundo por los desarrolladores. Es multiplataforma, y tiene multitud de interfaces de usuario. Además de permitirte mirar el código y descargar las diferentes versiones de una aplicación, la plataforma también hace las veces de red social conectando desarrolladores con usuarios para que estos puedan colaborar mejorando la aplicación.

La interfaz de usuario de GitHub es más fácil de usar que la de Git, lo que la hace accesible para personas con pocos o ningún conocimiento técnico. Esto significa que se puede incluir a más miembros del equipo en el progreso y la gestión de un proyecto, haciendo que el proceso de desarrollo sea más fluido.

Github permite que los desarrolladores alojen proyectos creando repositorios de forma gratuita. Pero hay que tener una cosa en mente, y es que para poder subir gratis los proyectos deberán ser de código abierto. Sino se quiere que la aplicación sea de código abierto, la plataforma también tiene una versión de pago para alojar proyectos de forma privada.

**Comandos**

1. git init: Crea el subdirectorio .git con archivos de Git para gestionar el repositorio
2. git add: Enviar del directorio de trabajo al stagin área, los cambios.
3. git commit: enviar del stagin área al repositorio local, los cambios realizados
4. git checkout: deshacer los cambios del repositorio local al stagin área
5. git reset: resetear los cambios del stagin área al directorio de trabajo
6. git status: saber el estado de los archivos
7. git diff: ver las diferencias entre el area staging y el de trabajo
8. git rm: borrar archivos del area staging y también lo borra del área de trabajo
9. git mv: mover renombrar archivos en el are staging y también lo borra del área de trabajo
10. git log: ver el listado de cambios realizados en el repositorio.

**Markdown**

Ver las siguientes paginas explicativas de la sintaxis para los markdown en Git:

1. <https://docs.github.com/es/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>
2. <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>

**Bibliografía:**

* B., G. (2023, 10 enero). Qué es GitHub y cómo empezar a usarlo. Tutoriales Hostinger. <https://www.hostinger.mx/tutoriales/que-es-github>
* Fernández, Y. (2019, 30 octubre). Qué es Github y qué es lo que le ofrece a los desarrolladores. Xataka. <https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores>
* De Diego, A. S. (s. f.). Git, GitHub y Markdown. <https://es.slideshare.net/asanzdiego/git-github-y-markdown>