Qué es Git

Git es una herramienta que realiza una función del control de versiones de código de forma distribuida, de la que destacamos varias características:

- 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
- Podemos movernos, como si tuviéramos un puntero en el tiempo, por todas las revisiones de código y desplazarnos una manera muy ágil.
- Es muy rápida
- Tiene un sistema de trabajo con ramas que lo hace especialmente potente
- En cuanto a la funcionalidad de las ramas, las mismas están destinadas a provocar proyectos divergentes de un proyecto principal, para hacer experimentos o para probar nuevas funcionalidades.
- Las ramas pueden tener una línea de progreso diferente de la rama principal donde está el Core de nuestro desarrollo. En algún momento podemos llegar a probar algunas de esas mejoras o cambios en el código y hacer una fusión a nuestro proyecto principal, ya que todo esto lo maneja Git de una forma muy eficiente

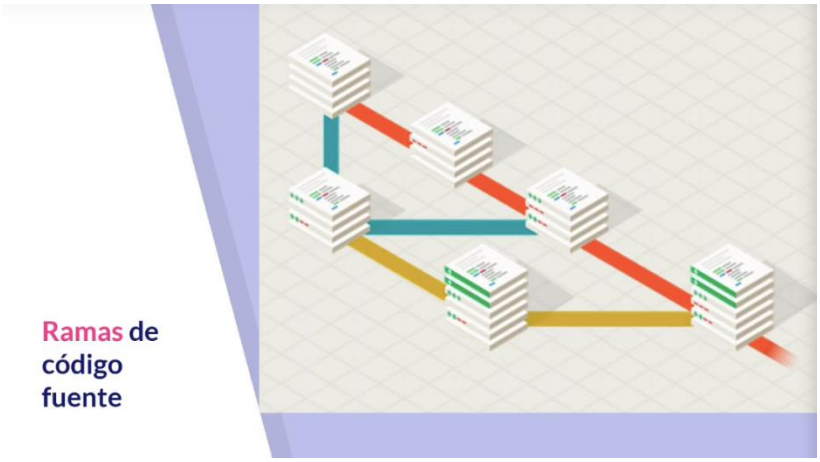

Figura 1. Ramas de codigo fuente. (Rubio, 2019)

Sistemas de Control de Versiones Distribuidos

Los Sistemas de Control de Versiones Centralizados (VCS), como por ejemplo Subversión, que es una herramienta en la que se ha confiado para albergar el histórico de revisión de versiones, es un punto centralizado, lo cual puede llegar a suponer una merma de trabajo si perdemos la conectividad de la red.

Los Sistemas de Control de Versiones Distribuidos (DVCS) salvan este problema. Algunos ejemplos de sistemas distribuidos, aparte de Git, son Mercurial, Bazaar o Darcs. En este tipo de herramientas, los clientes replican completamente el repositorio.

Funcionamiento de las réplicas

Las réplicas de versiones de proyectos en Sistemas de Control de Versiones Distribuidos funcionan como podemos ver en este gráfico:

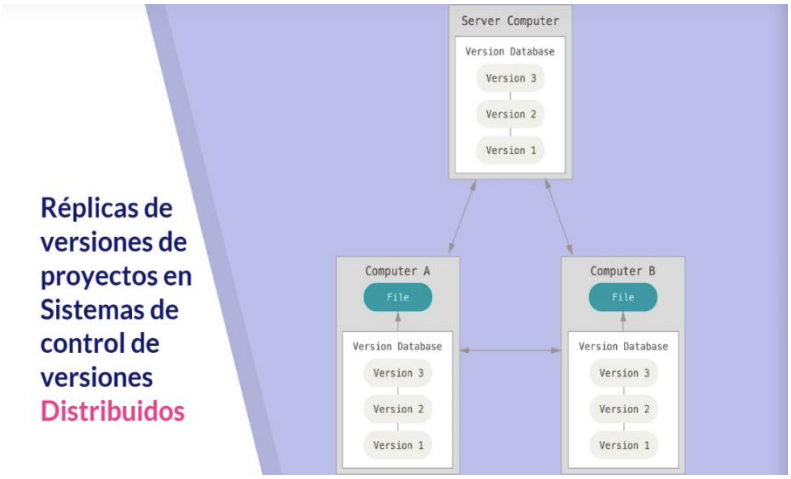

Figura 2. Réplicas de funciones. (Rubio, 2019)

Podemos tener una máquina que actúa como servidor, en el mismo se van almacenando las diferentes versiones de nuestro código, y cada uno de los clientes que participen en ese desarrollo tiene el histórico de las revisiones completa.

Sistemas de Control de Versiones Centralizados vs Sistemas de Control de Versiones Distribuidos

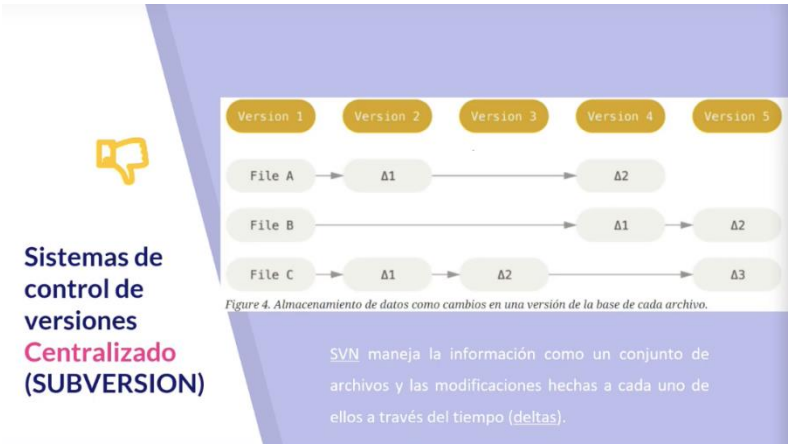

Figura 3. Sistemas de control de versiones centralizado. (Rubio, 2019)

En Sistemas de Control de Versiones Centralizados, como Subversión, partimos de una versión, por ejemplo de la versión 1 y tenemos tres ficheros, A, B y C.

De la versión 1 a la versión 2, las diferencias, como si fueran esa especie de incrementos que llamamos deltas, son almacenados por el sistema. De esta manera, por buscar algún símil, es como si Subversión trabajase con backups incrementales.

Después tenemos el funcionamiento de Sistemas de Control de Versiones de Git, en el que cada vez que hay cambios de ficheros éste es almacenado otra vez, y si no hay cambios es como si tuviésemos una especie de apuntador al fichero que no ha tenido cambios, en una revisión o en un hito del tiempo anterior.

Diferentes estados de un fichero Git

Existen tres tipos de estado de un fichero Git:

Secciones principales de un proyecto Git

- **Confirmado**: tenemos un fichero con el que hemos estado trabajando, hemos aprobado todos sus cambios y va en una nueva revisión, es decir, una especie de paquete dónde van todos los cambios.
- **Modificado:** estamos trabajando en el directorio de trabajo (o working directory) y consideramos que ese cambio debe de ir en una revisión, en esa especie de paquete, para formar la revisión.
- **Preparado:** hemos marcado un archivo para que vaya una revisión.

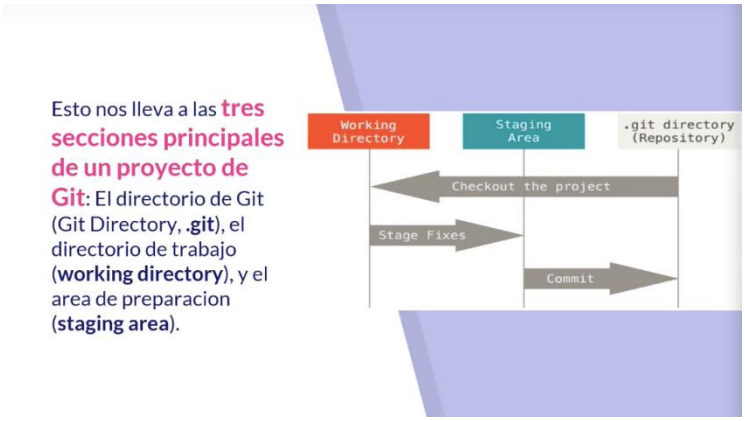

Figura 4. Seccione principales de un proyecto de Git. (Rubio, 2019)

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.

Referencias

Juan Carlos Rubio. (2019). Qué es GIT y para qué sirve. Recuperado de https://openwebinars.net/blog/que-es-git-y-para-que-sirve /