# Control de Versiones

Tal como lo expresa Roger Pressman (2010) en su libro, el control de versión combina procedimientos y herramientas para administrar diferentes versiones de objetos de configuración que se crean durante el proceso de software. Un sistema de control de versión implementa o se integra directamente con cuatro grandes capacidades:

1. Una base de datos de proyecto (repositorio) que almacena todos los objetos relevantes.
2. Capacidad de administración, de manera tal que se puedan almacenar todas las versiones de un objeto.
3. Facilidad para recopilar todos los objetos relevantes y construir una versión específica del software.

Por otro lado, Ian Sommerville (2011) define la gestión de versiones como el proceso de hacer un seguimiento de las diferentes versiones de los componentes de software o ítems de configuración, y los sistemas donde se usan dichos componentes. También incluye asegurar que los cambios hechos a dichas versiones por los diferentes desarrolladores no interfieran unos con otros. Por lo tanto, se puede considerar a la gestión de versiones como el proceso de administrar líneas de código y líneas base.

# Git

Git es un sistema de control de versiones distribuido (Distributed Version Control Systems - DVCS). En este tipo de sistemas los clientes no sólo descargan la última versión existente de los archivos sino que también realizan una réplica (copia de seguridad completa) del repositorio. De esta manera, si el servidor donde se encuentra alojado el repositorio deja de funcionar, mediante estas copias que poseen los usuarios es posible realizar una restauración completa.



Git fue desarrollado por la comunidad que desarrollaba el núcleo de Linux (y en particular a Linus Torvalds, el creador de Linux) persiguiendo los siguientes objetivos:

* Velocidad.
* Diseño sencillo.
* Fuerte apoyo al desarrollo no lineal (miles de ramas paralelas).
* Completamente distribuido.
* Capaz de manejar grandes proyectos (como el núcleo de Linux) de manera eficiente (velocidad y tamaño de los datos).

## Instantáneas, no diferencias

La principal diferencia entre Git y cualquier otro sistema de control de versiones radica en cómo Git modela sus datos. La mayoría de los sistemas almacenan información como una lista de cambios en los archivos (registran las modificaciones hechas sobre cada uno de ellos a lo largo del tiempo).



Git, en cambio, modela sus datos como un conjunto de *instantáneas*, conformando un sistema de archivos propio. Cada vez que un cambio es confirmado, Git realiza una “foto” (de aquí el término instantánea) de todos los archivos en ese instante de tiempo, y almacena una referencia a dicha instantánea.

Una característica que hace a la eficiencia de Git es que, si determinados archivos dentro del conjunto de cambios no se han modificado, Git no almacena dichos archivos nuevamente, sino que sólo mantiene un enlace al archivo anterior (idéntico).



## Operaciones locales

En Git, la mayoría de las operaciones solamente necesitan archivos y recursos locales (del cliente) para operar y por lo general, no se necesita información de ningún otro ordenador de la red.

## Integridad

Para garantizar esta propiedad, absolutamente todos los elementos son verificados mediante una suma de comprobación (checksum) antes de ser almacenados, y luego son identificados a partir de ésta.

Mediante esta implementación, Git asegura poder identificar claramente cualquier modificación realizada en los elementos del repositorio.

Esta funcionalidad está integrada en Git y asegura la no perdida de información durante la transmisión de datos a través de la red.

## Estados

Los archivos administrados por Git pueden adoptar tres estados:

* *Confirmado*: el archivo se encuentra almacenado de manera segura en la base de datos local.
* *Modificado:* el archivo ha sido modificado, pero aún no ha sido confirmado en la base de datos local.
* *Preparado:* el archivo modificado ha sido confirmado en su versión actual para ser incluido en la próxima confirmación de cambios.

## Secciones Principales

De esta manera, es posible definir las tres secciones principales de un proyecto en Git:

* *Directorio de Git (Git directory).*
  + Espacio donde Git almacena los metadatos y la base de datos de objetos para el proyecto del usuario. Éste es la parte más importante de Git puesto que se copia cuando se clona o replica un repositorio.
* *Directorio de trabajo (working directory).*
  + Copia de una determinada versión del proyecto (generalmente la última). Estos archivos son obtenidos a partir de la base de datos comprimida que se encuentra almacenada en el directorio de Git, y son copiados en el disco del usuario para que puedan ser utilizados y modificados.
* *Área de preparación (staging area)*
  + Se trata de un sencillo archivo, generalmente contenido en el directorio de Git, el cual almacena información referida a todas las modificaciones que serán incluidas en la próxima confirmación de cambios.

## Flujo de trabajo básico

* Modificación de una serie de archivos en el directorio de trabajo.
* Preparación de los archivos, añadiéndolos al área de preparación.
* Confirmación de los cambios, utiliza archivos que se encuetran en el área de preparación para almacenar las instantáneas correspondientes en el directorio de Git.



Tal como se puede observar en la figura inmediatamente anterior:

* Si una versión específica de un archivo se encuentra en el directorio de Git, se considera confirmada (**committed**).
* Si ha sufrido modificaciones desde se obtuvo del repositorio, pero ha sido añadida al área de preparación, se encuentra preparada (**staged**).
* Y si ha sufrido modificaciones desde que se obtuvo del repositorio, pero no se ha preparado, se encuentra modificada (**modified**).

## Actualidad

Desde su nacimiento en 2005, Git ha evolucionado y madurado para ser fácil de utilizar. Es extremadamente rápido y muy eficiente para grandes proyectos.

A continuación, se presentan algunas de las compañías que hacen uso de Git.



# Bibliografía

Chacon, S., & Straub, B. (2014 ). *Pro Git* (2nd ed.). Apress Berkely.

*Git*. (s.f.). Obtenido de https://git-scm.com/

Pressman, R. S. (2010). *Ingeniería del software - Un enfoque práctico* (Séptima ed.).

Sommerville, I. (2011). *Ingeniería de Software* (Novena ed.).