# Punto 3

## 1 – Herramientas disponibles en el mercado

Las herramientas disponibles en el mercado para realizar control de código fuente las podemos encontrar divididas en dos grupos.

Por un lado se encuentra el modelo cliente-servidor, el cual consta de un repositorio central al que se accede mediante un cliente instalado en la máquina local. Estos sistemas son:

**Subversion (svn)**: Es una de los sistemas de control de versión más utilizado. Se utiliza en proyectos como SourceForge, Apache, Python y Ruby, también es utilizado por Google Code para distribuir código. Para sistemas Windows, se suele utilizar la aplicación Tortoise SVN.

Subversion es desarrollado como un proyecto de la Apache Software Foundation y gracias a ello cuenta con una gran cantidad de desarrolladores y usuarios.

Ventajas:

* Gran comunidad de usuarios que lo utiliza.
* Integración con los principales entornos de desarrollo.
* Buena documentación.

Desventajas:

* Puede ser un poco lento.
* Se agrega bastante complejidad al manejar varios branchs.
* No cuenta con demasiados hosting gratuitos.

**Concurrent versions system (cvs)**: fue lanzado en 1986, es uno de los pioneros en este ámbito. Si bien se trata de una tecnología más antigua, sigue siendo útil para cualquier desarrollador.

Ventajas:

* Sistema muy simple.
* Consume pocos recursos.

Desventajas:

* Curva de aprendizaje no muy empinada.
* Tecnología en desuso.

Por otro lado encontramos el modelo distribuido:

**LibreSource**: está basado en Java, cuenta con herramientas visuales para facilitar el trabajo en equipo, lo cual lo diferencia de los otros proyectos los cuales son a nivel “línea de comandos”. A causa de esto no tiene una gran curva de aprendizaje. Se trata de una herramienta pensada para personas que no quieren aprender técnicas específicas y quieren centrarse más en la comunicación con los miembros del proyecto.

**Mercurial**: escrito en Python, pretende ser rápido, ligero, portable y fácil de usar. Fue diseñado para proyectos de gran tamaño, por lo tanto no fue pensado para desarrolladores independientes ni diseñadores. Su rendimiento y escalabilidad son una de las características más importantes. Es una herramienta simple de aprender Mercurial usa un protocolo eficiente, basado en HTTP, que persigue reducir el tamaño de los datos a transferir, así como la multiplicación de peticiones y conexiones nuevas. Mercurial puede funcionar también sobre ssh, siendo el protocolo muy similar al basado en HTTP.

Ventajas:

* Rápido y escalable.
* Soporta grandes proyectos.

Desventajas:

* Un poco costoso de aprender para principiantes.
* Comunidad no tan grande.

**Git** es una de las herramientas que está teniendo un rápido ascenso de los sistemas de control de versiones, teniendo un gran impacto para la comunidad de desarrollo web.

Comenzó gracias al núcleo de Linux que es un proyecto de software de código abierto con un alcance bastante importante. Durante los inicios de este, los cambios en el software se pasaron en forma de parches y archivos. Luego, el proyecto del núcleo de Linux empezó a usar controlador propietario llamado BitKeeper. La relación entre la comunidad que desarrollaba el núcleo de Linux y la compañía que desarrollaba BitKeeper se vino abajo, y la herramienta dejó de ser ofrecida gratuitamente. Esto impulsó a la comunidad de desarrollo de Linux (y en particular a Linus Torvalds, el creador de Linux) a desarrollar su propia herramienta basada en algunas de las lecciones que aprendieron durante el uso de BitKeeper.

Algunos de los objetivos del nuevo sistema fueron los siguientes:

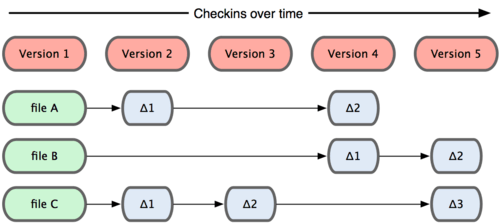
* Velocidad
* Diseño sencillo
* Desarrollo no lineal (miles de ramas paralelas)
* Completamente distribuido
* Capaz de manejar grandes proyectos de manera eficiente

Desde su nacimiento en 2005, ha evolucionado y madurado para ser fácil de usar y aún conservar estas cualidades iniciales. Es tremendamente rápido y muy eficiente con grandes proyectos.

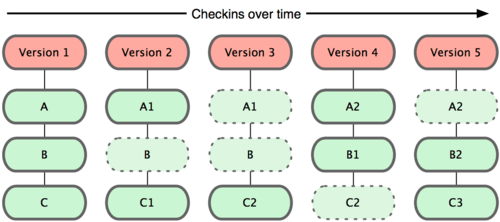
Git se enorgullece de ser un sistema rápido y eficaz, y muchos grandes proyectos de código abierto lo usan para alimentar sus repositorios; Proyectos como: Linux Kernel , VINO , Fedora.

***GitHub*** ha ayudado recientemente establecer Git como un gran sistema de control de versiones, ofreciendo una hermosa parte delantera para muchos proyectos grandes, como Rails y Prototype . Sin embargo, Git no es tan fácil de aprender como CVS o SVN, así que es mucho más difícil de usar para un principiante.

La principal diferencia entre Git y cualquier otro VCS (Subversion y compañía incluidos) es cómo Git modela sus datos. Conceptualmente, la mayoría de los sistemas almacenan la información como una lista de cambios en los archivos. Estos sistemas modelan la información que almacenan como un conjunto de archivos y las modificaciones hechas sobre cada uno de ellos a lo largo del tiempo. Tienden a almacenar los datos como cambios de cada archivo respecto a una versión base.



Git no modela ni almacena sus datos de este modo, los modela más como un conjunto de instantáneas de un mini sistema de archivos. Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en Git, él básicamente hace una foto del aspecto de todos tus archivos en ese momento, y guarda una referencia a esa instantánea. Para ser eficiente, si los archivos no se han modificado, no almacena el archivo de nuevo, sólo un enlace al archivo anterior idéntico que ya tiene almacenado. Se modela de la siguiente forma:

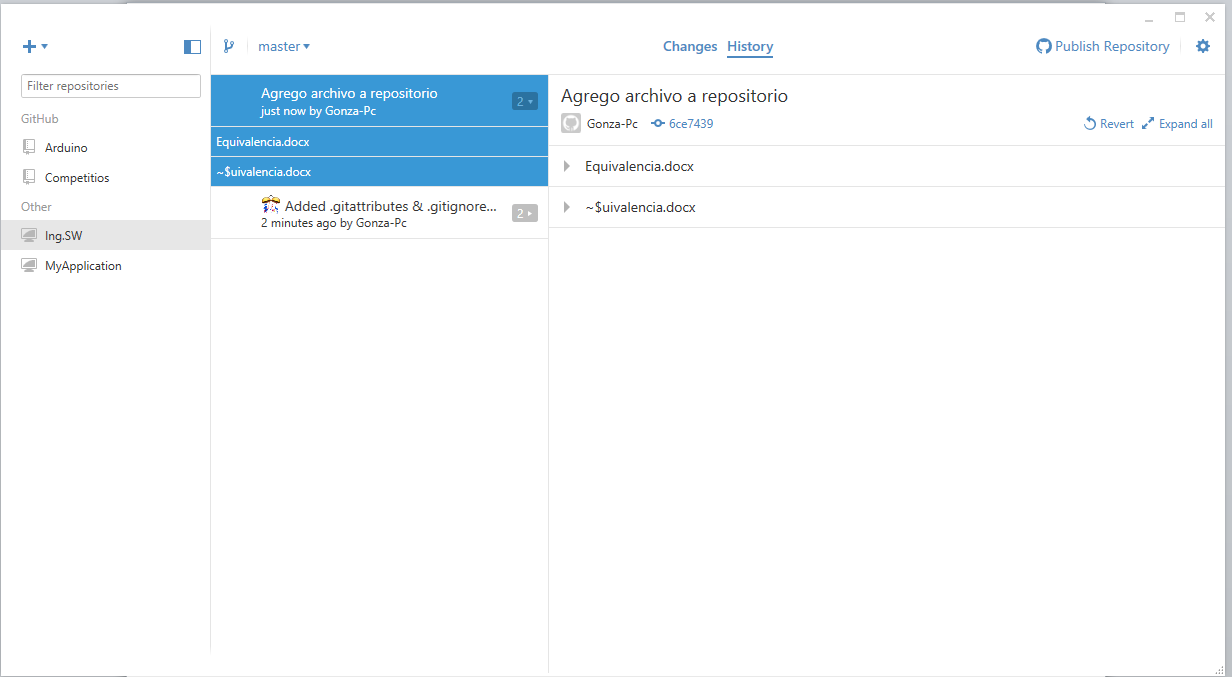


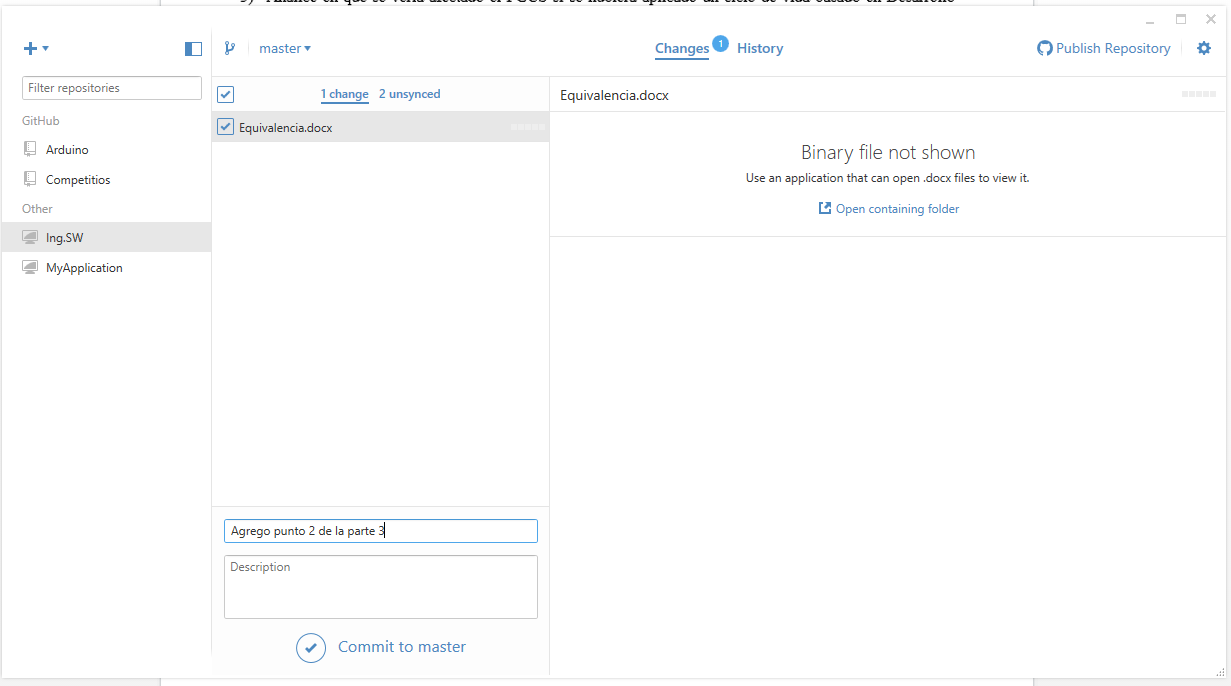
Almacena la información como instantáneas del proyecto a lo largo del tiempo.

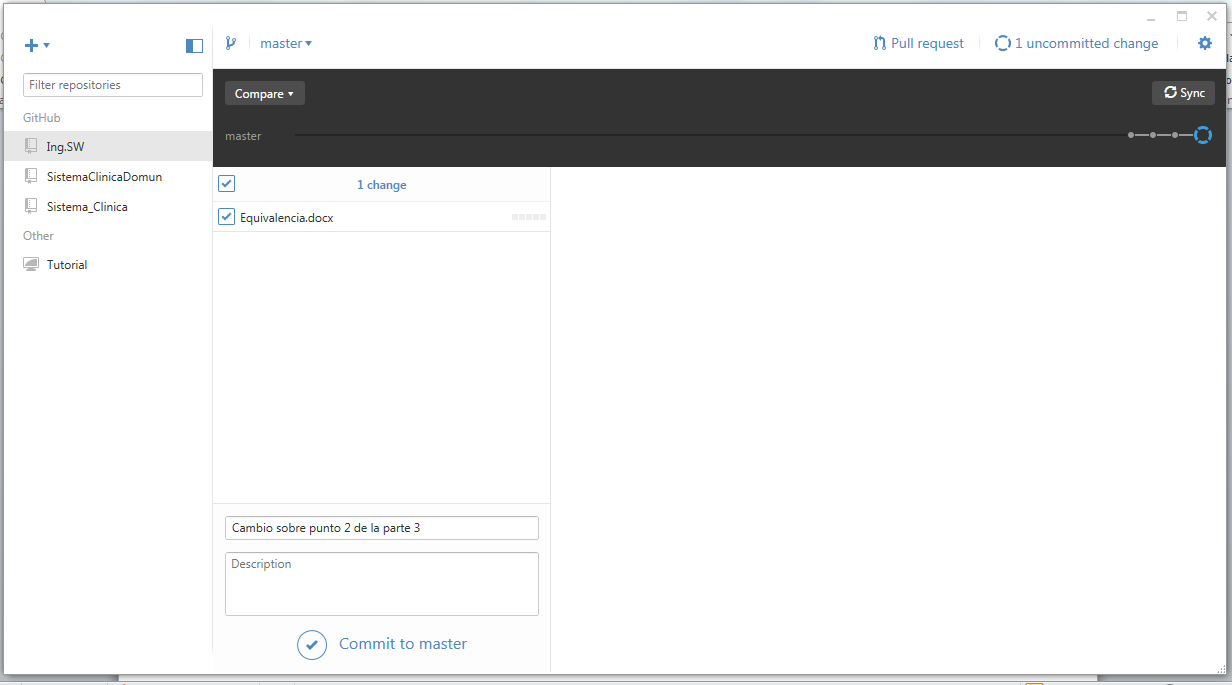
*Reconsidera casi todos los aspectos del control de versiones que muchos de los sistemas copiaron de la generación anterior. Esto hace que se parezca más a un mini sistema de archivos con algunas herramientas tremendamente potentes construidas sobre él, que a un VCS.*

Alguna de las desventajas que podemos encontrar es que no hay muchos entornos de desarrollo integrados, aunque esto va disminuyendo ya que los principales entornos empiezan a estar integrados. Otra desventaja puede ser que se requiera un poco de documentación a la hora de empezar a utilizarlo, ya que hay algunos conceptos nuevos en él.

2 –







4 - Desde un punto de vista práctico podemos decir que el uso de un controlador de código fuente trae muchos beneficios al ser utilizados. Entre estos podemos encontrar:

* + poder trabajar varias personas sobre un mismo módulo de la aplicación, facilitando el trabajo entre los distintos participantes.
  + poder llevar un historial y auditoría sobre el código fuente.
  + posibilidad de volver a versiones anteriores con gran facilidad.
  + permite realizar distintas ramas de código, posibilitando que se trabaje sobre distintas funciones de una misma aplicación en ambientes distintos y poder unificar las cuándo se desee.

Probando agregar cambios en repositorio en un archivo existente y con otro usuario.