**VERSIONAMIENTO DE SOFTWARE**

El control de versiones de software es un sistema mediante el cual las diferentes versiones de un programa de software en particular se numeran tanto para uso interno como para designación de versiones. Este sistema permite a los desarrolladores de software rastrear más fácilmente los cambios entre versiones y permite a los clientes y usuarios del software reconocer mejor las versiones actualizadas.

Por lo general, se usa un sistema numérico, a menudo con un decimal para indicar actualizaciones de versiones menores. Las indicaciones de versiones de software han entrado en un uso popular y en el léxico principal debido a la importancia de este sistema para permitir a los usuarios de software solucionar problemas y encontrar actualizaciones de software.

Los programas que uno puede encontrar en la web pueden venir en cinco estados: pre-alpha, alpha, beta, Release Candidate y Final Release.

* **Pre-alfa**: a veces es el primero en salir. Un programa que aún no es completamente funcional con las características que constituirán su versión final. A veces ni siquiera tiene todas las funciones que eventualmente tendrá.
* **Alfa:** esta es la versión para probadores de software. Los software testers (personas que prueban el software) suelen ser personas que pertenecen a la empresa que desarrolla el software o a la comunidad que lo rodea. En el software libre y de código abierto, se acostumbra a distribuir versiones alfa del software y su código para que cualquier persona interesada pueda contribuir informando de errores. Es un software en estado alfa que, si bien tiene todas sus características, solo funciona parcialmente y generalmente es inestable, inseguro o ambos.
* **Beta**: El software actual tiene todas sus características y parece casi la versión definitiva. Puede seguir siendo un poco inestable pero no es inseguro. La idea es corregir los errores que puedan aparecer, que suelen ser de menor gravedad. Hay algunos programas que tardan mucho en pasar a la versión beta y pueden continuar haciéndolo durante mucho tiempo porque aún no cumplen con los requisitos del desarrollador para su versión final. Sin embargo, normalmente se pueden utilizar sin problemas graves, dependiendo del programa.
* **Release Candidate (RC):** puede llamarse 'Candidate Release', 'Final Candidate' o simplemente 'Candidate'. Será prácticamente igual que en la versión final, a menos que haya errores que no se detectaron en las pruebas anteriores. Por lo general el software está libre de errores que pueden impedir su desarrollo.
* **Ultimate Edition:** también conocida como Commonly Available o Dorada (Gold). Las versiones finales no vienen con nombres como las versiones anteriores. Como máximo, un nombre de marca (XP, en Windows) o un número que indique la versión en la que se encuentra (3, en Firefox).

Otro sistema es el sistema de numeración que según la posición de los números (XX.YY.ZZ) muestra el avance del proyecto, así como las distintas actualizaciones que ha sufrido indicando el tipo e importancia

Cada uno de estos números tienen el siguiente significado:

* **MAYOR(XX):** solo es editado cuando las nuevas características en el desarrollo a ser propuestas como publicación conllevan a que se puedan ocasionar fallos debido nuevas formas de implementar los métodos, eliminación de funciones o incluso a cambios de nombres de estas.
* **MINOR(YY):** es editado a cada publicación que exprese nuevas características, que conllevan a mejorar y ampliar tu proyecto, sin afectar las funciones existentes.
* **BUGFIX(ZZ):** es editado en cada corrección que haces en producción, osea los cambios que realizas para enmendar errores de publicaciones anteriores.
* **Entrega**: este dígito tiene el objetivo de llevar la cuenta del número de veces que una entrega se rechaza, por incumplimiento de algún requisito de la gestión de entregas o del proyecto.

En el caso de la creación de un branch o linea de desarrollo distinta de la principal añadiremos la letra b al final de la numeración.

Además de estos existen sistemas que no siguen el versionado tradicional.

Algunos proyectos, como [Ubuntu](https://es.wikipedia.org/wiki/Ubuntu), usan los dígitos para indicar la fecha de lanzamiento. Otros proyectos, como en arcades, usan códigos varios, finalizando en una fecha y la versión del parche.

En algunos proyectos de desarrollo ágil, como [Google Chrome](https://es.wikipedia.org/wiki/Google_Chrome), se usa un versionado similar al tradicional. Aunque los cambios en el dígito mayor indican cambios relevantes en el software, no son tantos como es habitual en el versionado tradicional.

**Ejemplos**

**Con fechas**

Año.mes: 15.4

Año.mes.menor: 15.4.2

Año.mayor.menor: 2015.1.2

**Usado en arcades**

Código:Idioma:Mueble:Modificador:AñoMesDíaParche: ABC:S:A:0:2021111501

**¿Qué es un Milestone 1 o M1?**

Un hito (milestone) es un evento significativo que tiene importancia para todo el proyecto. Se utiliza para medir el progreso y mostrar cómo se acerca a los objetivos que se han establecido.

Esta breve definición de hitos en la gestión de proyectos puede parecer bastante simple. Sin embargo, tenga en cuenta que pueden haber algunas confusiones. Por ejemplo, los hitos pueden ser confundidos fácilmente con tareas, objetivos y entregables.

**La importancia de hitos en la gestión de proyectos**

Ya que sabemos que es un hito en un proyecto, vamos a averiguar por qué es importante que los proyectos incluyan hitos.

Los hitos proporcionan muchos beneficios. Las siguientes ventajas son las más esenciales.

* Los hitos te permiten conocer todos los plazos
* Los hitos ayudan a identificar cuellos de botella y posibles retrasos
* Los hitos ayudan a crear un plan sencillo

**¿Cuáles son los ejemplos de hitos de un proyecto?**

Cada proyecto e industria tendrá diferentes hitos. Al mismo tiempo, hay algunos ejemplos de hitos importantes del proyecto que se pueden establecer como una forma de mantener a su equipo en el camino correcto.

Para sistematizarlo de alguna manera, dividiremos los ejemplos de hitos en tres grupos:

* Hitos de la etapa inicial.
* Hitos de medio término del proyecto.
* Hitos de las etapas finales.

Ejemplos de hitos de un proyecto en la etapa inicial

* Aprobación del proyecto
* Financiación
* Aprobación de diseño

Ejemplos de hitos de un proyecto en medio término

* Finalización de tareas críticas
* KPI alcanzados: Los KPI son los indicadores clave de desempeño. Representan objetivos medibles que ayudan a realizar un seguimiento del desempeño general de los proyectos.
* Recolección de comentarios

Ejemplos de hitos de un proyecto en las etapas finales

* Pruebas
* Aprobación final

**Snapshot**

El snapshot se le conoce de muchas formas y en ocasiones se confunde con otros sistemas de seguridad como las backups. Las snapshot, también denominadas instantáneas de volumen o VSS (volumen snapshot service) son un elemento de seguridad informática complementarias a las copias de seguridad o backups. Es un mecanismo usado entre otras tareas para proteger el desarrollo de un proyecto creando instantáneas de momentos concretos del desarrollo del software evitando así la posible pérdida de datos.

Hay muchas diferencias de una copia de seguridad, con una snapshot por lo que sus usos, aunque complementarios con una [Backup](https://www.ymant.com/servicios-informaticos/seguridad/), son de extrema utilidad. Aunque nunca se recomienda una VSS como único sistema de seguridad para tus datos.

**¿Que son los VSS?**

Las instantáneas de volumen son copias de un instante determinado de los archivos, permitiendo recuperar las carpetas y los archivos tal y como estaban en el momento que se estableció dicha instantánea. Estas trabajan sobre el origen de los datos, haciendo bloques y añadiéndole a esos bloques nuevos datos. De esta forma cuando se quiere volver al momento que tomamos esa “imagen” se descartarían los datos que se añadieron sobre esos bloques de datos.

Pueden ser muy útiles para momentos puntuales como realizar un cambio importante en tu equipo o servidor. Ya que si sale mal dispondrás de una copia exacta que poder ejecutar para volver a la normalidad rápidamente.

Cabe destacar que siempre es importante contar con un backup ya que son mas seguros, esto es debido a que las [Copias de seguridad](https://www.ymant.com/servicios-informaticos/seguridad/) se hacen en otra fuente de almacenamiento distinto a los archivos del sistema, También las copias de seguridad son programables por lo que puedes renovarlas habitualmente.

Las snapshots por el contrario se hacen de forma manual y trabajan directamente sobre la fuente de los datos y hace una copia del estado de los archivos en un momento concreto. Pero a diferencia de los backups pueden corromperse o haber errores, que provocaran la perdida de toda o parte de los datos y se nos hará imposible volver al momento donde hicimos el snapshot.

**Las ventajas de un snapshot**

Las ventajas vienen dadas de la velocidad de replicación y la facilidad de clonar servidores o máquinas.

La velocidad de restaurar un servidor o equipo con un VSS o snapshot es de segundos ya que se encuentra en el origen de los datos, mientras que un backup es un proceso más lento (esto está motivado a que los datos están en un dispositivo externo y el tiempo de restauración variara en función a la velocidad de transmisión)

Son muy convenientes para mejorar el tiempo de respuesta en por ejemplo el caso de que se haya producido un cambio importante o un error al actualizar un software. Habiendo hecho el snapshot se volverá al momento exacto antes de que se produjera ese fallo.

Son rápidos de crear y tienen un bajo impacto sobre la producción.

Son perfectos para servidores que necesitan tener una [alta disponibilidad](https://es.wikipedia.org/wiki/Copia_instant%C3%A1nea_de_volumen) de un servidor y no es posible pararlo para hacer un backup, de esta forma no se interrumpe el servicio.

Resumiendo, los snapshot o VSS son el complemento perfecto a las copias de seguridad para hacer más ágil la recuperación ante una pérdida de datos.