


## VERSIONAMIENTO SEMÁNTICO

Existe en el desarrollo de software actual, en el que se utiliza un innumerable número de frameworks, micro frameworks, librerías y otras hierbas, surge la pesadilla de mantener una gestión de dependencias entre unas librerías con otras.

Por eso existe una convención para etiquetar las versiones de software que nos conviene mantener. En este post comento de forma resumida en qué consiste dicha convención, llamada SemVer.

**La sintaxis de versión es:**



MAJOR . MINOR . PATCH

- 1) PATCH: Cuando se producen cambios menores, típicamente arreglos de bugs que en ningún momento interfieren en la compatibilidad con la versión anterior.
- 2) MINOR: Cuando se introducen nuevas funcionalidades o unas mejoras notables sobre las funcionalidades anteriores. Tampoco interfiere en la compatibilidad.
- 3) MAJOR: Gran cantidad de cambios, o aun siendo pocos cambios, éstos son muy significativos, abandono de soporte a funcionalidades deprecated. Puede interferir, y de hecho lo suele hacer, en la compatibilidad con versiones anteriores.

Así que si en nuestro proyecto actualizamos a nueva versión mayor es necesario realizar una batería completa de pruebas. Se pueden utilizar los gestores de descargas de librerías como bower o composer para tener controlada y automatizada la actualización de librerías.

### Distintos tipos de versiones

El versionamiento semántico es un convenio o estándar a la hora de definir la versión de tu código, dependiendo de la naturaleza del cambio que estás introduciendo. De tal forma, se identifican 3 tipos de cambios:

**Major:** Cambio drástico en el software. No es compatible con código hecho en versiones anteriores.

**Minor:** Cambio que añade alguna característica nueva al software o modifica alguna ya existente, pero que sigue siendo compatible con código existente. También cuando marcamos algo como obsoleto.

**Patch:** Cuando arreglamos un bug siendo el cambio retrocompatible.

De esta forma, tenemos un lenguaje común entre desarrolladores y consumidores a la hora de hablar de versiones.

## ¿Cómo se marca una versión como major, minor o patch?

Cada vez que enviemos código al repositorio crearemos un nuevo tag siguiendo el convenio semántico, dependiendo de los cambios introducidos. El tag contendrá la información, separando las versiones de cada tipo por puntos, de la forma `major.minor.patch`. Esta versión en concreto, el tag que lo representa, no puede ser cambiado jamás, para que si alguien depende de esa versión en particular, pueda seguir haciéndolo sin problemas.

Si el framework o librería que yo utilizo sigue el versionamiento semántico a rajatabla, sé que puedo actualizar sin miedo todas las versiones marcadas como patches o minor porque no romperán mi código. Yo podría actualizar de la versión 5.3.3 a la 5.3.4 o incluso 5.4.0, porque solo estarían incrementando las versiones minor y patch, que son compatibles con código existente.

Si por el contrario pasase de la versión 1.4 a la versión 2.0, tendría que tener cuidado porque seguramente habría muchas cosas que dejarían de funcionar: ha cambiado la major.

## Identificadores de estabilidad

Además de poder definir los cambios en el código como major, minor o patch, se suelen añadir unos identificadores que ayudan a marcar versiones específicas que quieres diferenciar, indicando la estabilidad de esa versión.

Por ejemplo, tienes tu aplicación a la versión 1.4.6 pero decides empezar el desarrollo de la próxima gran versión, la 2. Entonces los primeros desarrollos en esa versión nueva irán a la 2.0.0, pero como todavía estás empezando y probando cosas, podrías ponerle un identificador que le dijese a la gente la estabilidad de esa versión. Tu versión quedaría en algo como 2.0.0-alpha, por ejemplo, y la gente sabría que es una versión `_alpha_` no muy estable.

Según fueses avanzando en el desarrollo, llegarías a una versión beta la cual marcarías por ejemplo como 2.1.3-beta. Así hasta que todo estuviese listo para publicarse, y marcas una versión como candidata para ser publicada como 2.1.5-rc1.

Si dos versiones iguales tienen distinto identificador, da igual: se consideran iguales a todos los efectos. Esto nos permite ir actualizando la estabilidad de una versión, pudiéndola pasar de `_alpha_` a `_beta_` o incluso a release candidate.

También por convenio, encontrarás otro identificador conocido como dev-master. Este identificador apunta a la última revisión enviada al repositorio, que no tiene por qué ser la última estable. Si dependes en esta versión estarás al día con todos los últimos cambios, pero te arriesgas a recibir cambios no retrocompatibles.