**VERSIONADO SEMÁNTICO**

Al hablar de software una palabra muy recurrente es “dependencias” y se sabe que entre más grande sea el sistema más dependencias habrá. Para trabajar con un proyecto con estas características el artículo propone un conjunto de reglas a seguir para versionar, y a este sistema se le llama “Versionado semántico”, además este sistema da sentido al proyecto.

Hay algunas palabras clave: DEBE, NO DEBE, OBLIGATORIO, DEBERÁ, NO DEBERÁ, DEBERÍA, NO DEBERÍA, RECOMENDADO, PUEDE y OPCIONAL. Se interpretan de acuerdo a lo propuesto en el siguiente link: <https://www.rfc-es.org/rfc/rfc2119-es.txt>

**Especificación del Versionado Semántico**

1.- El software DEBE declarar un API público comprehensivo y preciso.

2.- Un número de versión normal DEBE tener la forma de X.Y.Z donde X, Y y Z son números enteros no negativos, y NO DEBEN ser precedidos de ceros. X es la versión mayor, Y es la versión menor, y Z es la versión parche. Cada elemento DEBE incrementarse numéricamente. Por ejemplo: 1.9.0 -> 1.10.0 -> 1.11.0

3.- El paquete ya versionado y publicado NO DEBE ser modificado. Si hay modificaciones se debe publicar como una nueva versión.

4.- Una versión mayor en cero (0.y.z) se considera como desarrollo inicial. Todo PUEDE cambiar en cualquier momento.

5.- La versión 1.0.0 define el API público.

6.- La versión parche Z (x.y.Z x > 0) DEBE ser incrementada si solamente se introducen correcciones de errores compatibles con versiones anteriores. Una corrección de error se define como un cambio interno que corrige un comportamiento incorrecto.

7.- La versión menor Y (x.Y.z x > 0) DEBE ser incrementada si se introduce funcionalidad nueva y compatible con la versión anterior del API público. Ésta DEBE ser incrementada si se introduce cualquier funcionalidad al API público o mejora al código privado. Este PUEDE incluir cambios a nivel de parches. La versión parche DEBE reiniciarse a 0 cuando una versión menor se incrementa.

8.- La versión mayor (X.y.z X > 0) DEBE ser incrementada solamente si se introducen cambios incompatibles con la versión anterior del API público. Este PUEDE incluir cambios de nivel menor y parches. Versiones parche y menores DEBEN ser reiniciadas a 0 cuando una versión mayor es incrementada.

9.- Una versión de prelanzamiento PUEDE ser denotada agregando un guión y una serie de identificadores separados por puntos, inmediatamente seguida de la versión parche.

10.- Metadatos de compilación PUEDEN ser denotados agregando el signo más y una serie de identificadores separados por puntos, inmediatamente seguido de la versión parche o prelanzamiento.

11.- La precedencia se refiere a cómo las versiones se comparan entre ellas cuando se ordenan.

El versionado semántico da el beneficio de ahorrar tiempo al proporcionar una manera de publicar y actualizar paquetes sin tener que crear nuevas versiones de paquetes dependientes.