Una sugerencia práctica de versionado semántico

20230107T2230

Una sugerencia práctica de números de versión semánticos: X.Y.p.c

(nota: Está basada en el estándar https://semver.org/lang/es/, solo que usando la parte de “parche” para ”pruebas” de desarrollo.)

Cara a los usuarios finales, solo existirán las dos primeras cifras X.Y

=> X es versión mayor: un lanzamiento completo “a bombo y platillo”.

=> Y es versión menor: un lanzamiento de correcciones, compatible con cualquier otra con el mismo X.

También lo podemos ver como que:

=> Aumentamos X cuando se emite un ‘lanzamiento’ completo a producción.

=> Aumentamos Y cuando se emite una ‘corrección’ sobre la última versión X en producción.

nota: La emisión de una versión X.Y siempre va con p0 y c0 => vX.Y.p0.c0

Cara al desarrollo interno, existirán las cuatro cifras: X.Y.p.c

=> pN.c0 es la versión N de pruebas,
una versión que solo verán los usuarios internos de pruebas: vX.Y.p1.c0, vX.Y.p2.c0, etc.

=> pN.c1es una versión de trabajo, preparando la emisión de pN+1,
una versión que solo verán los programadores: vX.Y.p2.c1, etc.

También lo podemos ver como que:

=> Aumentamos p cuando se emite una versión ‘para pruebas’ internas.

=> Aumentamos c cuando empezamos a trabajar después de cualquier emisión.

nota: La emisión de una versión de prueba siempre va con c0 => vX.Y.pN.c0

Ejemplo práctico:

	DESARROLLO	PRODUCCIÓN	notas
trabajando	1.0.0.1		
emisión de una versión para probar	v1.0.p1.c0		
trabajando	1.0.1.1		
emisión de una versión para probar	v1.0.p2.c0		
	etc		
emisión de una versión para usuarios		**v1.1**.p0.c0	

trabajando	2.0.0.1		
emisión de una versión para probar	v2.0.p1.c0		
trabajando	2.0.1.1		
trabajando en la corrección de un bug		1.1.0.1	se parte desde la 1.1.0.0
para probar la corrección		v1.1.p1.c0	
trabajando en la corrección de un bug		1.1.1.1	
para probar la corrección		v1.1.p2.c0	
emisión de una corrección para usuarios		**v1.2**.p0.c0	
incorporar la corrección al desarrollo en curso	2.0.1.1		(*)
trabajando	2.0.1.1		(*)
emisión de una versión para probar	v2.0.p2.c0		
	etc		
emisión de una versión para usuarios		**v2.1**.p0.c0	Se retira la **v1.Y**.p0.c0 que estuviera activa en ese momento.
trabajando	3.0.0.1		
emisión de una versión para probar	v3.0.p1.c0		
trabajando	3.0.1.1		
emisión de una versión para probar	v3.0.p2.c0		
trabajando	3.0.2.1		
emisión de una versión para probar	v3.0.p3.c0		
	etc		
		etc	

Nota: (*) Incrementar .c más allá de .1 es opcional, ya que es algo solo para la persona o el equipo que está programando.

IMPORTANTE:

Una vez lanzada una versión X.Y (`vX.Y.p0.c0`) o X.Y.p (`vX.Y.pN.c0`),

JAMÁS de los jamases SACAR OTRA CON EL MISMO X.Y.p

=> Al ponerse a trabajar en la siguiente: incrementar .c (poner al menos X.Y.p.1)

=> En el siguiente lanzamiento: incrementar .p (si es de pruebas) o X.Y (si es de producción)

nota: No pasa nada por lanzar una X.Y para después, en muy poco tiempo, lanzar otra X.Y+1 e inmediatamente después otra X.Y+2 (por ejemplo, si se detectan errores graves y se han de corregir con rapidez)…
Lo importante es que cada versión lanzada sea unívocamente identificable.

Tampoco pasa nada por llegar hasta versiones del estilo de v57.24.p8.c0 o similares...

EL VERSIONADO ES SOLO PARA IDENTIFICAR
y saber con qué pieza ÚNICA de software estamos lidiando en cada momento.

Cuando se está diseñando un esquema de versionado. Conviene no mezclar aspectos "estéticos" o "marquetinianos" en ese diseño. Procurar no empezar a considerar temas tales como qué número de versión queda más o menos "bonito", o cual es más o menos "vendible", o cual revela menos la cantidad de errores que se han tenido que corregir, o...

La mejor manera de mantener la unicidad de versiones es seguir reglas objetivas, simples y claras.