El versionado semántico es un sistema utilizado para asignar números de versión a los software de manera que refleje los cambios que se hacen en la API pública de forma clara y predecible. Este sistema utiliza tres números separados por puntos (X.Y.Z) para indicar los cambios principales (X), cambios menores (Y) y correcciones de errores (Z), lo que facilita a los desarrolladores y usuarios entender cómo estos cambios afectan al software. Asimismo, algunos desarrolladores emplean cuatro dígitos, siendo el cuarto lo que se llama número de compilación.

**Especificaciones del versionado semántico:**

1. **Declarar una API pública:** Todo software que usa versionado semántico debe declarar su API pública, que puede estar dentro del código o en la documentación. Esta declaración debe ser precisa y completa para que los desarrolladores sepan cómo interactuar con el software.
2. **Formato de la versión:** El número de versión debe seguir el formato X.Y.Z, donde X es la versión mayor, Y es la versión menor y Z es la versión de parches. Estos números deben ser enteros sin ceros iniciales y siempre deben aumentar de forma secuencial (por ejemplo, 1.9.0 -> 1.10.0 -> 1.11.0).
3. **No modificar versiones publicadas:** Una vez que una versión ha sido lanzada, no debe ser modificada. Si se necesitan realizar cambios, estos deben ser lanzados como una nueva versión.
4. **Versión mayor cero (0.y.z):** Mientras un software esté en la versión 0.y.z, está en desarrollo inicial. Todo puede cambiar sin previo aviso, y no se debe considerar la API pública como estable.
5. **Versión 1.0.0:** La versión 1.0.0 define la primera API pública estable. A partir de este punto, cualquier cambio en la versión se reflejará en cómo cambia la API pública.
6. **Incremento de versión de parches:** El número Z (parches) debe incrementarse solo si se corrigen errores que no rompen la compatibilidad hacia atrás. Un parche es cualquier corrección interna que soluciona comportamientos incorrectos.
7. **Incremento de versión menor:** El número Y (versión menor) debe incrementarse cuando se agregue funcionalidad nueva que no rompa la compatibilidad hacia atrás. También debe incrementarse si se depreca alguna funcionalidad en la API pública. Además, puede incluir cambios de nivel de parche, y el número Z se reinicia a 0 cuando Y se incrementa.
8. **Incremento de versión mayor:** El número X (versión mayor) debe incrementarse cuando se introduzcan cambios en la API pública que no sean compatibles con versiones anteriores. También puede incluir cambios menores y de parches, y tanto Y como Z se reinician a 0 cuando se incrementa X.
9. **Versiones preliminares (pre-release):** Una versión preliminar puede indicarse añadiendo un guion y una serie de identificadores (como “alpha” o “beta”) después de la versión de parche. Estas versiones son inestables y pueden no ser totalmente compatibles con versiones anteriores. Por ejemplo, 1.0.0-alpha es una versión preliminar.
10. **Metadatos de compilación:** Los metadatos de compilación pueden agregarse añadiendo un signo de más (+) seguido de una serie de identificadores. Estos identificadores se usan para información adicional y no afectan el orden de precedencia de las versiones. Por ejemplo, 1.0.0+build20130313 no tiene prioridad sobre 1.0.0.
11. **Precedencia de versiones:** La precedencia indica cómo se comparan las versiones. Se determina al comparar los números de versión mayor, menor y de parches en ese orden. Si estos son iguales, las versiones preliminares tienen menor precedencia que las versiones normales. Dentro de las versiones preliminares, la precedencia se determina comparando cada identificador (numéricos se comparan como números y los alfanuméricos de manera léxica).

**19-09-2024**