8 JUL, 2020

Versionamento semântico – O que é e como usar

EDSON AMARAL JR (HTTPS://IMASTERS.COM.BR/PERFIL/EDSONAM
Tem 1 artigos publicados com 100 visualizações desde 2020

EDSON AMARAL JR (HTTPS://IMASTERS.COM.BR/PERFIL/EDSONAMARALJR)

1

Edson Amaral é desenvolvedor de Software há 15 anos. Já atuou nos setores financeiro, tecnologia e serviços, construindo soluções para as áreas financeiras, comércio, transporte aéreo entre outros. Hoje faz parte do time de desenvolvimento da Dotz.

LEIA MAIS (HTTPS://IMASTERS.COM.BR/PERFIL/EDSONAMARALJR)

Resumidamente e de forma objetiva o versionamento semântico ou Semantic Versioning é um conjunto de regras e requerimentos para atribuição de versão de software. Em outras palavras, é uma tentativa de resolução de um problema antigo conhecido como “inferno de dependências” ou dependency hell que nada mais é do que problemas e complicações ao lidar com pacotes de softwares.

Esse problema ocorre quando uma versão de software prejudica a segurança e integridade de uma nova versão impedindo assim que o projeto prossiga.

O versionamento semântico garante um controle da versão do código utilizando um grupo de números mantendo assim sua compatibilidade e integridade nas novas publicações conforme exemplo abaixo:

Versionamento semântico 1.0.0

– 1 (Major) – controle de compatibilidade. Informa que existem funcionalidades/códigos incompatíveis com as versões anteriores.

– 0 (Minor) – controle de funcionalidade. Informa que novas funcionalidades foram adicionadas ao código.

– 0 (Patch) – controle de correção de bugs. Informa que um ou mais erros foram identificados e corrigidos.

– Pré-release – versão candidata. É uma versão com algumas instabilidades pois pode ter incompatibilidades no pacote.

Como usar

Pensando em como evitar que as novas atualizações quebrem o software já publicado foram elaboradas 11 regras chave que compõem essa especificação que quando seguidas de forma correta garantam a estabilidade do pacote publicado. Por exemplo, uma dessas regras obriga que uma versão específica do software seja "imutável", ou seja, após o lançamento de uma versão sob hipótese alguma essa versão poderá ser excluída ou modificada, independente da existência de bugs ou melhoria de alguma funcionalidade existente.

As regras

Primeira Regra – API pública

No documento original de versionamento semântico é definido que qualquer software que esteja utilizando versionamento semântico "deve" definir uma API pública, seja em sua documentação ou declarado em seu código-fonte e que a existência da mesma pode representar diversos fatores, dentre eles:

→
Disponibilidade de bibliotecas, ou seja, classes, métodos e funções que possam ser utilizados por outros softwares.

→
Caso seu software não seja feito para uso público, alguma outra forma de interação com os usuários do mesmo deve existir.

→
A existência de endpoints na API.

Em resumo uma API pública dependerá exclusivamente de seu objetivo e forma de interação utilizada.

Segunda Regra – Formato do número de versão

Essa regra define basicamente o que já foi descrito no início do artigo. Obrigatoriamente o versionamento semântico deve ter seu formato de versão conforme x.y.z onde, todos devam ser números inteiros e não negativos e não deve existir zeros à sua esquerda, ou seja, a versão semântica deverá sempre iniciar com 1.

Além do mais cada bloco deverá ter seu nome onde "x" será a versão principal ou maior, "y" a versão secundária ou menor e "z" a versão de correção ou patch.

Terceira Regra – Versões imutáveis

"Once a versioned package has been released, the contents of that version MUST NOT be modified. Any modifications MUST be released as a new version."

O documento original é bem claro quanto a essa regra onde uma vez determinada e disponibilizada, a versão do software não poderá ser modificada em hipótese alguma. Assim sendo, caso ocorra qualquer necessidade de correção de bug, modificações, atualizações ou nova funcionalidade deverá ser lançada uma nova versão.

Quarta Regra – Desenvolvimento inicial

No desenvolvimento inicial de um software sua primeira versão deverá ser setada como zero, isso deve-se ao fato de que muita coisa poderá mudar a qualquer momento antes do seu lançamento e com isso sua versão inicial não pode e nem deve ser considerada estável.

Em outras palavras, um software com sua versão inicial definida poderá mudar a qualquer hora porém, não quer dizer que versões anteriores ou dependências não poderão ser utilizadas em projetos. Entretanto, versões futuras deverão ter seu ponto de atenção mesmo quando sejam versões secundárias ou correções (patches).

Quinta Regra – Versão 1.0.0

Uma API pública poderá ser considerada “definida” a partir da versão 1.0.0 ou seja, versões futuras deverão ser baseadas ou dependentes dessa versão ou conforme ela é modificada.

Assim sendo, uma versão 1.0.0 por exemplo, nunca poderá ser baseada em uma versão 0.8.0 da API pública. Porém versões superiores deverão de alguma forma ser dependentes da versão 1.0.0.

Sexta Regra – Versão de remendo ou patch version

Essa regra define que a versão de remendo, correção ou patch deverá indicar a correção de um ou mais bugs que se mantém compatível com a versão anterior. Ou seja, uma versão 1.0.1 corrige um problema na versão 1.0.0 e será totalmente compatível com a mesma.

Sétima Regra – Versão secundária ou minor version

O conceito de versão secundária mostra que a correção está sendo lançada relacionada com o versionamento anterior de correção (patch). Essa versão secundária apenas poderá ser implementada sob as seguintes condições:

→
Quando forem adicionadas novas funcionalidades, porém ainda assim compatíveis com as versões anteriores;

→
Quando houver depreciação de alguma funcionalidade já publicada;

→
Quando houverem novas funcionalidades ou melhorias introduzidas ao código.

Além disso sempre que houver o lançamento de uma versão secundária para correção de bugs de versões anteriores a numeração da versão de remendo deverá ser zerada. Por exemplo, um projeto lançado na versão 1.0.6 que teve sua versão secundária publicada para correção de algum bug será setado para 1.1.0 e não 1.1.6;

Oitava Regra – Versão principal ou major version

A oitava regra para versionamento semântico é definida conforme o documento oficial de versionamento semântico e explicada originalmente como:

“Major version X (X.y.z | X > 0) MUST be incremented if any backwards incompatible changes are introduced to the public API. It MAY include minor and patch level changes. Patch and minor version MUST be reset to 0 when major version is incremented”.

Em outras palavras, sempre que alterações incompatíveis com versões anteriores forem implementadas na versão principal da API pública as versões menores e de alterações deverão ser resetadas para 0 (zero)

Nona Regra – Versão de pré-lançamento ou pré-release

A nona regra trata de versões de pré-lançamento e explica que:

Uma versão de pré-lançamento será indicada quando um hífen e uma séries de identificadores separados por pontos estiverem presentes logo após a versão do patch. Esses identificadores deverão conter apenas caracteres alfanuméricos ASCII e hífen (0-9A-Za-z). Além disso os identificadores NÃO DEVEM estar vazios e NEM incluir zeros à sua esquerda.

Embora não seja obrigatório nos números de versão padronizados por versionamento semântico, podem ajudar em algumas situações.

Décima Regra – Dados de build

A décima regra é apenas mais uma regra adicional, portanto pode ser classificada como opcional. Segundo essa regra, é possível, se necessário adicionar informações extras de build junto à versão utilizando o caractere “+” seguido de informações adicionais separadas por pontos e que contenham apenas caracteres alfanuméricos e hífen. Podem ser adicionados como dados de build por exemplo o timestamp de quando o build foi criado ou até mesmo um hash do commit ou arquivos de build.

Décima Primeira Regra – Precedência de versões

Na décima primeira regra é falado sobre algo muito importante e utilizado por gerenciadores de dependências como o Bower e o Composer por exemplo. Em resumo essa regra define a ordem em que as versões do versionamento semântico deverá ter. No caso do versionamento semântico essa ordem será numérica (1.0.0 < 2.0.0 < 3.0.0 < 3.1.0) e assim por diante. Além disso essa ordem poderá contar com algumas características adicionais.

→
cada identificador deverá ser comparado da esquerda para a direita;

→
identificadores compostos serão comparados numericamente;

→
identificadores contendo letras ou hífens serão comparados lexicalmente conforme a organização da tabela ASCII onde A < a < b

→
Caso os identificadores estejam misturados entre numéricos e não-numéricos os não-numéricos terão precedência aos numéricos;

→
Uma versão com maior quantidade de identificadores terá precedência a uma versão com menor quantidade de identificadores, ou seja, 1.0.0-alpha < 1.0.0-alpha.1;

Conclusão

Imagine se essas regras chave não existissem? Seria uma imensa bagunça controlar os pacotes já publicados e suas alterações não é mesmo? Além do mais o versionamento é apenas mais um dos diversos processos necessários que podem ser utilizados em controle e distribuição de softwares para facilitar os times.

Referências:

https://semver.org/#spec-item-1 (https://semver.org/#spec-item-1)

https://fjorgemota.com/versionamento-semantico-ou-como-versionar-software/ (https://fjorgemota.com/versionamento-semantico-ou-como-versionar-software/)

https://www.desenvolvedormatteus.com.br/versionamento-semantico-2-0-0/
(https://www.desenvolvedormatteus.com.br/versionamento-semantico-2-0-0/)

https://engineering.contaazul.com/versionamento-de-software-na-era-%C3%A1gil-8b53f6c08192
(https://engineering.contaazul.com/versionamento-de-software-na-era-%C3%A1gil-8b53f6c08192)

SAIBA MAIS
(HTTPS://IMASTERS.COM.BR/PERFIL/EDSONAMARALJR)

Edson Amaral Jr
(mailto:edsonamaraljr@inmasters.com.br)

1 Artigo(s)

Edson Amaral é desenvolvedor de Software há 15 anos. Já atuou nos setores financeiro, tecnologia e serviços, construindo soluções para as áreas financeiras, comércio, transporte aéreo entre outros. Hoje faz parte do time de desenvolvimento da Dotz.

Este projeto é mantido e patrocinado pelas empresas

(https://apiki.com/)

(https://asaas.com/)

(https://code.iadb.org/pt)

(http://www.dialhost.com.br)

(https://fullcycle.com.br)

(https://huaweicloud.imasters.com.br/)

(https://www.idexo.com.br/)

(https://carreiras.itau.com.b
utm_campaign=topimais&ut

(https://nearsure.com/)

(https://www.onlyoffice.com/pt/)

(https://dev.paygo.com.br/)

(https://superviz.com/)

(https://developers.totvs.com/)

(https://wake.tech/)

ASSINE NOSSA
Newsletter

Fique em dia com as novidades do iMasters! Assine nossa newsletter e receba conteúdos especiais curados por nossa equipe

Qual é o seu e-mail?

ASSINAR