# Versionando seu projeto

## Como versionar corretamente o seu projeto e assim conseguir informar rapidamente que tipo de mudança ocorreu.

## Introdução

E aí pessoal, tenho ficado distante pacas, fazendo só posts dos links lidos na semana, mas hoje resolvi escrever alguma coisa, pouca e básica, mas não menos importante. Vou tentar fazer assim, para conseguir criar mais conteúdo e também me animar/agitar um pouco. Nesse exato momento estou ouvindo uma playlist chamada [Melancholia](https://open.spotify.com/playlist/37i9dQZF1DWZrc3lwvImLj?si=T01Fdb5ASAa8yHHpnAUNJQ), que tem músicas bem bacanas e que no momento está combinando um pouco comigo...

## O que é versionar e por que versionar?

Antes mesmo de tudo, o que seria versionar um projeto? Todo projeto bem organizado possui um escopo, onde definimos certas metas a alcançar e para cada uma dessas metas, podemos criar uma espécie de release, que seria nossa versão.

Outro motivo para se versionar é que assim conseguimos "guardar" vários estágios daquele projeto e podemos continuar andando com o desenvolvimento. Assim as pessoas conseguem usar a versão que melhor lhes caber. É o que fazemos em geral com a maioria das bibliotecas e frameworks que utilizamos.

## Usando o Semantic Version

Para manter um padrão mais organizado para versionar, nasceu um projeto chamado também de [SemVer](http://semver.org/), que é um sistema bem básico. Ele é controlado pelo formato x.y.z:

* x : fica para as famosas Major Versions, que seriam as versões principais fechadas.
* y: fica para as Minor Versions, que seriam adições de pequenas melhorias.
* z: fica para os patchs, que seriam correções de bugs.

Então, se tivéssemos, por exemplo, a versão 1.4.11, significaria que ela já teve sua primeira versão lançada, 4 novas features adicionadas depois e 11 correções.

## Entendendo como versionar

Digamos que eu tenha uma simples API com um método all que retorna um array contendo todos os dados que eu desejo e essa é minha primeira versão estável da API, ela então terá a versão 1.0.0.

Depois de lançado, eu vi que dentro do meu método all, tinha um bug que fazia não retornar o último item do array. Eu apliquei a correção e lancei uma nova versão. Como era somente um bug, eu deverei mudar o valor da última casa e por isso a versão será 1.0.1.

Mais para frente eu resolvi criar um método chamado getItem(item), que é responsável por pegar um único item daquele array todo. Como eu adicionei uma nova feature, eu irei mudar o número do meio e resetar o último dígito, assim a versão ficará como 1.1.0, pois irei criar uma nova versão minor e essa ainda não teve nenhuma correção de bugs.

Se um dia eu resolver mudar as assinaturas dos métodos, modificando seus nomes e as formas de chamar, então será uma grande mudança e para isso eu irei alterar o primeiro número e por esta ser uma nova versão, os outros números serão resetados e com isso eu terei a versão 2.0.0.

## Usando as tags do git para versionar

Eu aqui falei só na teoria como nomear e versionar, mas como fazer isso na real? Usando o git é muito simples, basta usarmos as tags.

O git permite dois tipos de tags que são annotated e non-annontated. A annotated tag permite guardar mais informações sobre aquela tag em si, como comentários e meta-informação.

Para criar uma tag basta:

* git tag 1.0.0 : isso criará uma non-annontated tag que vai referenciar ao commit atual.
* git tag -a 1.0.0 -m 'Mensagem da Tag' : isso irá criar uma annotated tag, contendo uma mensagem informativa e uma referência para o objeto do commit.

Para listar todas as tags criadas, basta usar git tag. Se quiser apagar alguma tag use o comando git tag -d 1.0.0.

Lembre-se sempre de usar git push --tags para subir as tags criadas para o seu repositório remoto e também de usar git fetch --all para sincronizar as tags no seu local.

Caso você queira ver uma versão específica, use git checkout 1.0.0 e estará exatamente onde aquela versão parou, isso se torna extremamente útil.

## Ferramentas para facilitar

Uns dias atrás, fazendo umas video-aulas muito legais sobre [Como escrever uma biblioteca javascript opensource](https://egghead.io/lessons/javascript-how-to-write-a-javascript-library-automating-releases-with-semantic-release) eu acabei encontrando uma ferramenta bem bacana para ajudar no versionamento, se chama [semantic-release-cli](https://github.com/semantic-release/semantic-release), vale dar uma olhada tanto no curso quanto na ferramenta =)

## Conclusão

Bom, a ideia desse post é fazê-los terem essa visão de que versionamento é importante e se ele for bem feito, só de ler os números de versões, já conseguimos entender se algo muito grande que mudou ou se algo foi adicionado ou corrigido. Passem a criar versões para suas libs e passem a fazê-las do modo certo, o mundo agradece.

https://willianjusten.com.br/semantic-version/