**Convenção para versionamento Git**

A forma de organização de branchs foi inspirado no post publicado no blog do Fernando Jorge Mota. A organização e o processo para nomeação de branchs proposta no artigo do Fernando Jorge utiliza o GitFlow, mas no processo implementado por mimnão será utilizado o GitFlow para controlar o envio de arquivos ao repositorio, esse processo sera feito através das ferramentas (plug-ins) fornecidos pelo Eclipse para o Git.

**Git Flow – Uma forma legal de organizar repositórios git**

Uma dúvida muito comum a quem começa a [usar o Git](http://fjorgemota.com/git-sistema-de-controle-de-versoes-distribuido/) de maneira mais ativa é como organizar as branches, afinal, são muitos os problemas que um projeto pode enfrentar: De bugs urgentes que devem ser corrigidos, a criação de inúmeras features em conjunto com *releases* agrupando os deploys relativos a essas features. Mas..como organizar tudo? Pensando nisso é que foi criado o Git Flow, um modelo de organização de branches criado por Vincent Driessen que mais tarde se tornou uma excelente extensão ao Git, permitindo seu uso de forma fácil com qualquer repositório git, e é sobre isso que vou falar hoje.

Antes de começar a falar mais detalhadamente sobre o **Git Flow**, quero chamar atenção para o fato de que o **Git Flow** não é o único modelo de organização de branches: Existem outros por aí,  e inclusive existem empresas que adotam modelos de organização próprios, mas, disso tudo, um fato é que o **Git Flow** parece ser o mais conhecido e usado, se encaixando muito bem numa plenitude de situações, e sendo muito bom para trabalhos em equipe (por experiência própria). Se você conhecer alguma alternativa que funciona bem, [mande nos comentários](https://fjorgemota.com/git-flow-uma-forma-legal-de-organizar-repositorios-git/#comments)!

Agora, continuando as explicações sobre o **Git Flow**, como explicado no primeiro parágrafo, trata-se de um modelo de organização de branches desenvolvido especialmente para o git. Por ser primariamente um modelo de organização de branches, isso significa que o Git Flow estabelece algumas regras de **nomenclaturas** para tipos de branches enquanto, ao mesmo tempo, define o que **cada tipo de branch faz.** Para referência, segue uma lista dos tipos de branches definidos pelo Git Flow e suas respectivas descrições:

* **Branch master** – É a branch que contém código em nível de **produção**, ou seja, o código mais maduro existente na sua aplicação. Todo o código novo produzido eventualmente é juntado com a branch **master**, em algum momento do desenvolvimento;
* **Branch develop** – É a branch que contém código em nível preparatório para o próximo deploy. Ou seja, quando features são terminadas, elas são juntadas com a branch **develop**, testadas (em conjunto, no caso de mais de uma feature), e somente depois as atualizações da branch **develop**passam por mais um processo para então ser juntadas com a branch **master**;
* **Branches feature/\*** – São branches no qual são desenvolvidos recursos novos para o projeto em questão. Essas branches tem por convenção nome começando com **feature/** (exemplo: **feature/new-layout**) e são criadas a partir da branch **develop** (pois um recurso pode depender diretamente de outro recurso em algumas situações), e, ao final, são juntadas com a branch **develop**;
* **Branches** **hotfix/**\* – São branches no qual são realizadas correções de bugs críticos encontrados em ambiente de produção, e que por isso são criadas a partir da branch **master**, e são juntadas diretamente com a branch **master**e com a branch **develop** (pois os próximos deploys também devem receber correções de bugs críticos, certo?). Por convenção, essas branches tem o nome começando com **hotfix/** e terminando com o próximo sub-número de versão (exemplo: **hotfix/2.31.1**), normalmente seguindo as regras de algum padrão de versionamento, como o [padrão do versionamento semântico, que abordei neste post](http://fjorgemota.com/versionamento-semantico-ou-como-versionar-software/);
* **Branches release/\*** – São branches com um nível de confiança maior do que a branch **develop**, e que se encontram em nível de preparação para ser juntada com a branch **master**e com a branch **develop** (para caso tenha ocorrido alguma correção de bug na branch **release/\*** em questão). Note que, nessas branches, bugs encontrados durante os testes das *features* que vão para produção podem ser corrigidos mais tranquilamente, **antes** de irem efetivamente para produção. Por convenção, essas branches tem o nome começando com **release/** e terminando com o número da próxima versão do software (seguindo o exemplo do *hotfix*, dado acima, seria algo como **release/2.32.0**), normalmente seguindo as regras do versionamento semântico, como falado acima;