Fluxo de trabalho Git flow

O Gitflow foi idealizado por Vincent Driessen em 2010 e trata-se de um padrão de gerenciamento de fluxo de trabalho que tem como base o Git. Seu modelo de ramificações possui uma estrutura bem definida, com funções próprias e regras de interação. Na prática, esse rigor garante organização e controle para o gerenciamento de grandes projetos.

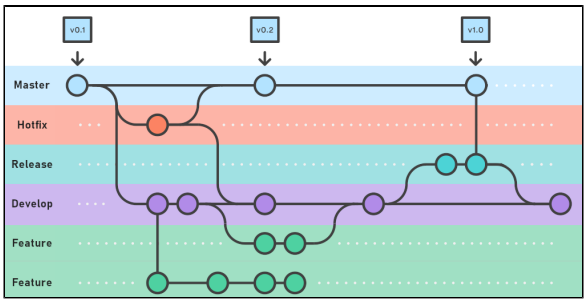

Figura 1 - Linha do tempo em um projeto com o padrão Gitflow

Ramificações

Para gerenciar o fluxo de trabalho com o Gitflow não é necessário novas tecnologias, conceitos ou comandos, apenas os recursos já disponíveis no Git, como *branch* (ramo), *merge*, *checkout*, *tags* etc.

O Gitflow possui 5 tipos de ramos, dois deles são considerados principais (*master* e *develop*) e os demais secundários ou temporários (*release*, *feature* e *hotfix*/*bugfix*). Veremos a seguir a definição de cada um deles.

Principais

O ramo *master* é responsável por manter o histórico de todas as versões de lançamento (produção) do projeto, enquanto o *developer*, que é derivado do *master* assim que o repositório é criado, possui o histórico completo do produto. Eles seguirão em paralelo, de forma contínua e nunca serão mesclados diretamente durante toda a vida do projeto, assim, o *develop* sempre utilizará um ramo secundário (*relea se*) para atualizar o *master*.

O ramo *develop* reflete o estado mais atual das funcionalidades que estão prontas (desenvolvidas) e foram elegíveis para ir para a próxima *release*, já o *master* possui o conteúdo um pouco mais defasado, porém mais seguro para o usuário do sistema.

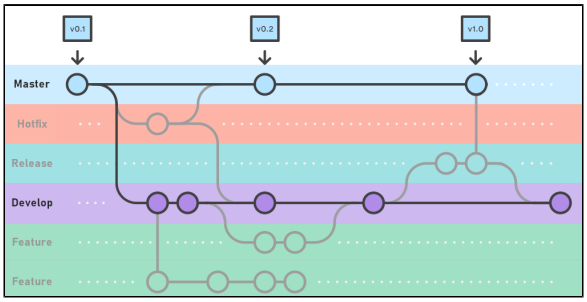

Figura 1.1 - Destaque para os ramos principais: master e develop

Novos recursos

Para cada nova funcionalidade a ser criada no sistema será gerado um ramo *feature* próprio a partir da última versão do *develop*, o ramo mais estável em desenvolvimento. O tempo de vida do novo ramo poderá variar de acordo com o tamanho da nova funcionalidade, porém, após finalizado será mesclado com o *develop* e excluído em seguida.

Caso seu projeto possua um gerenciador de tarefas, como Jira ou Redmine, é uma boa prática criar os ramos do tipo *feature* com o nome dos tickets, ex.: `feature/<número do ticket>`, facilitando a identificação e objetivo do ramo.

Nesta página

Conteúdo complementar

- Versionamento usando Git
- Solução de conflitos do Git

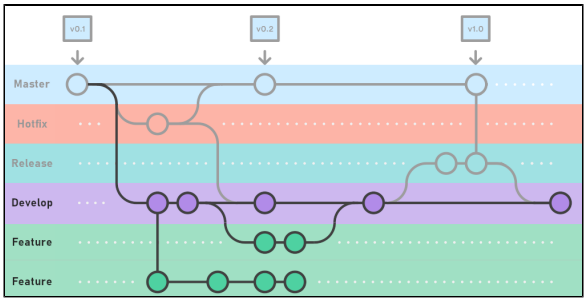

Figura 1.2 - Os branches feature só interagem com o develop

Lançamentos

Assim que o ramo *develop* estiver pronto para um lançamento, seja por acúmulo de novos recursos ou data pré-definida, um ramo *release* deve ser gerado a partir do *develop*, dando início a um novo ciclo de lançamento. Por serem consideradas versões "fixas", os ramos de lançamento são isolados em um ambiente de testes e novos recursos não devem ser adicionados a eles, apenas correções de *bug*, geração de documentação e outras tarefas relacionadas ao lançamento.

Na etapa final de lançamento, o ramo *release* será mesclado tanto com o *master*, dando origem a uma nova versão de produção, quanto com o *develop*, já que correções de *bugs* e pequenos ajustes foram realizados durante o *release*. Ao final dos dois *merges*, o ramo *release* é excluído.

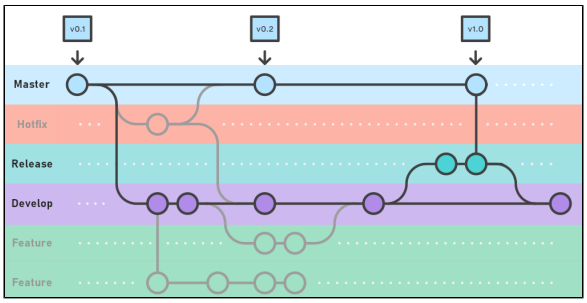

Figura 1.3 - Fluxo do lançamento de versão

Normalmente os nomes dados aos ramos do tipo *release* seguem um padrão `release/<versão>`, com versões incrementais, ex.: 1.0 (*major version*), 1.0.1 *(minor version*). O Gitflow também adota a prática de utilizar *tags* para marcar os *releases* após irem ao *master*, então, após fazer o *merge* de um *release* para o *master*, uma *Tag* com o número da versão pode ser criada para marcar o ponto (*commit*) do *deploy*.

Manutenções

Os ramos de manutenção (*bugfix* e *hotfix*) são utilizados para corrigir *bugs* ou problemas críticos que não podem esperar o próximo ciclo de lançamento.

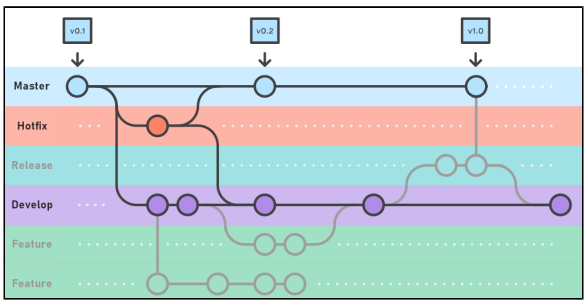

Figura 1.4 - Fluxo do hotfix

- **bugfix**: se um *bug* é encontrando durante a etapa de testes no *release*, é criado um branch *bugfix*, a correção é feita, ocorre um merge com o ramo *release* e o ramo de correção é excluído em seguida.
- **hotfix**: criado ao encontrar um problema crítico que necessita de uma ação imediata no *master*. Assim, após gerar um ramo *hotfix* a partir do *master* e corrigir o problema, serão realizados *merges* no *master*, com criação de uma nova *tag*, e no *developer*. Ao final, o ramo *hotfix* é excluído.

A nomenclatura usada para o *hotfix* e *bugfix* seguem o seguinte padrão: `hotfix/<versão>` e `bugfix/<versão>`, onde <versão> é o valor da nova *tag* aplicado em *master* ou *release* após mesclar o ramo.

Git flow no Cronapp

O Cronapp possui algumas funcionalidades que facilitam o trabalho com o Gitflow, veremos aqui como utilizar.

Caso tenha dúvidas sobre Git, recomendamos olhar primeiro a nossa documentação Versionamento usando Git.

Iniciar

Após criar e exportar seu projeto para algum repositório remoto, será possível acessar as opções do menu **Equipe** no menu do sistema. Assim, acesse **Equipe > Git flow > Iniciar Git flow**.

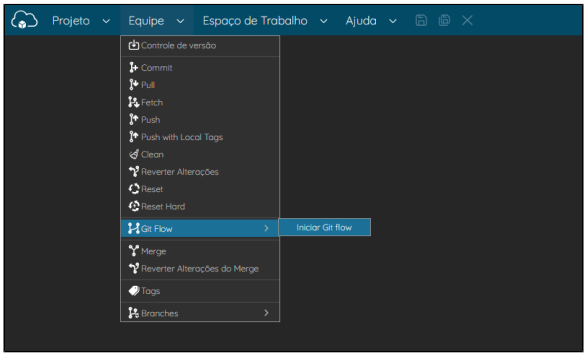

Figura 2 - Iniciando o Git flow em um projeto no Cronapp

Após essa ação, o Cronapp criará um ramo *develop* a partir do master e disponibilizará as opções para criar os seguintes ramos: *feature, release* e *hotfix*.

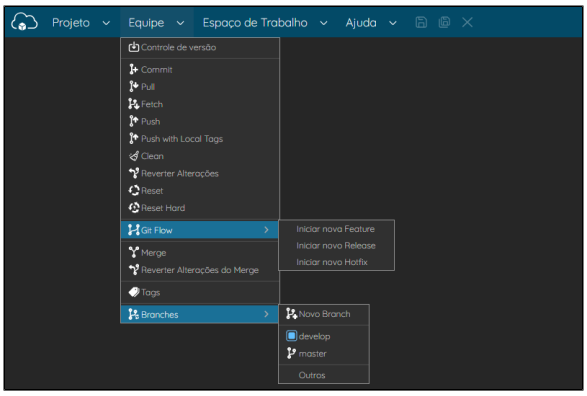

Figura 2.1 - Atualizações no projeto local após iniciar o Gitflow

Iniciar ramo

Ao solicitar um novo ramo a partir do submenu **Git flow** (Figura 2.1), uma janela de configuração do novo ramo será aberto.

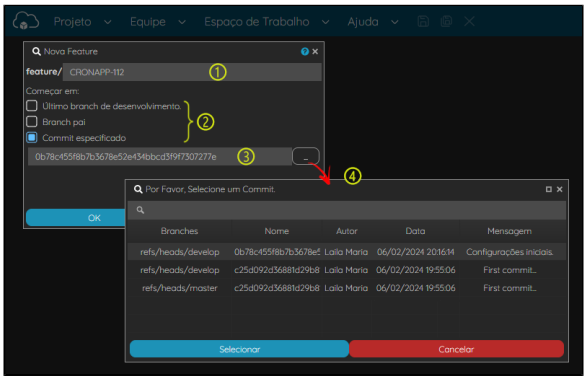

Figura 2.2 - Configurando novos branches no Git flow

1. **Nome do ramo**: os nomes dos ramos criados a partir do Gitflow seguem o padrão "<tipo de branch>/<nome>"
    - **<tipo de ramo>**: definido a partir do ramo selecionado *feature*, *release* ou *hotfix*.
    - **<nome>**: conteúdo informado no campo.
2. **Começar em**: aqui é possível selecionar de que ramo será bifurcado o novo ramo.
    - **Último branch de desenvolvimento**;
    - **Branch pai**;
    - **Commit especificado**: habilita o campo de seleção de *commit*.
3. **Campo**: habilitado ao selecionar a opção "Commit especificado", clique no botão "..." para abrir a janela.
4. **Janela de seleção de commit**: exibe todos *commits* dos ramos locais do projeto, inclusive os gerados pelo Git flow.

Finalizar ramo

Sempre que o ramo ativo tiver sido criado a partir do Gitflow, o submenu **Equipe** só exibirá a opção **Finalizar *hotfix*/*release*/*feature***.

- **Hotfix**: atualiza os ramos *master* e *develop*;
- **Release**: atualiza os ramos *master* e *develop*;
- **Feature**: atualiza o ramo no qual foi originado.

Ao final, é perguntado se deseja excluir o ramo (Figura 2.3).

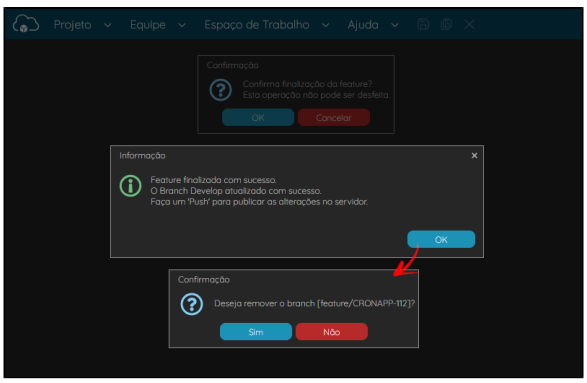

Figura 2.3 - Finalizando um branch Gitflow

* As figuras dos fluxos estão sob licença Creative Commons Atribuição 2.5 da Austrália, foram obtidas da Atlassian e modificadas.