07

O que aprendemos?

Nesta aula, aprendemos:

- Que é uma convenção bem seguida que a *branch* `master` tenha apenas os *commits* prontos para ir para produção;
- Que não é interessante realizar trabalho e *commitar* diretamente na *branch* `master` ;
- Como remover uma *branch*:
  - `git branch -d {nome_branch}` remove uma *branch* que já tem seu trabalho unido à *branch* atual;
  - `git branch -D {nome_branch}` remove uma *branch* mesmo que os *commits* desta *branch* ainda não estejam na *branch* atual, ou seja, força a remoção;
- Um pouco do processo chamado de ***Git Flow***:
  - Entendemos que o estado do código representado pela *branch* `master` deve ser o mesmo que estará em produção
  - Vimos que deve haver uma *branch* de desenvolvimento (comumente chamado de `develop`), onde todas as funcionalidades e correções devem ser muito bem testadas antes de ir para produção (`master`)
  - Vimos que cada funcionalidade deve ser feita em uma *branch* separada, e que é comum que esta *branch* tenha `feature/` como prefixo
  - Aprendemos também que *bugs* normalmente são corrigidos em *branches* separadas, com o prefixo `hotfix/`
  - Além disso, *branches* específicas para cada *release* são criadas para realizar os testes e correções de *bugs* específicos