Git e Github: Estratégias de ramificação, Conflitos e Pull Requests

- Esse curso capacita a:
    - Aprender estratégias de branching como Git Flow;
    - Usar técnicas avançadas como cherry-pick ou Bisect;
    - Conhecer ferramentas visuais como Git Desktop;
    - Trabalhar com Hooks e Eventos;
    - Entender o Pull Request;

Aulas:

1. **Github e OpenSource:**
    - Issues do Github, servem para reportar problemas, sugerir melhorias no código, solicitar novas funcionalidades e organizar quaisquer coisas que façam sentido para o projeto;
    - Para contribuir com um projeto no Github:
        1) Fazer Fork;
        2) Sincronizar o repositório local com o remoto;
        3) Feita as alterações, da push para o remoto, e em seguida fazer um pull request para o proprietário do repositório fazer uma verificação do código criado e se tudo estiver correto, mergiar;
    - **git rebase –i <hashDoCommitAnteriorAoQueVaiMudar>;**
    - Alternativas ao Github: Bitbucket [Vantagem: quantos projetos privados desejar], GitLab;

2. **Controle avançado de conflitos:**
    - **git cherry-pick <hash>** [Traz um commit especifico de outra branch para a branch em que se encontra];
    - **git bisect start** [Inicia um sistema de busca entre determinados commits]
      git bisect bad <hashDoCommitQueTáRuim>, dá Enter, depois
      git bisect good <hashDoUltimoCommitBom>, dá Enter, em seguida ele vai ficar mostrando como o codigo esta entre esses commits, e é preciso ficar avaliando como good ou bad e termina com **git bisect reset**;
    - **git show <hash>** [Mostra todas as alterações feita neste commit];
    - **git blame <file>** [Serve para ver quem é o responsável por cada linha no código];

3. **Estratégias de branching:**
    - **git branch –d <nome>** ou **git branch –D <nome>** [Servem para deletar um branch, e o com "-D" se o branch estiver avançado em relação ao atual];
    - Git Flow:
        master – Deve ser o mesmo que estará em produção;
        hotfix – Bugs normalmente são corrigidos aqui (hotfix/);

release – São criadas para realizar os testes e correções de bugs específicos;

Develop – Onde todas as funcionalidades e correções devem ser testadas antes de ir para produção;

Feature – Cada funcionalidade deve ser feita em um branch separado (feature/);

- 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 os 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;

4. **Ferramentas visuais:**
   - Que há ferramentas visuais que podem nos auxiliar com o trabalho com o Git;
   - O **Git Cola** foi uma das primeiras ferramentas visuais multiplataforma. Embora não seja a mais complexa ou visualmente atraente, é bem completa e pode nos ajudar bastante;
   - O **GitHub Desktop** pode ser interessante para gerenciar os projetos do GitHub de forma mais ágil e facilitada, sem a necessidade de acessar o site;
   - O **GitKraken** é uma ferramenta extremamente completa, que nos auxilia inclusive com a implementação do Git Flow;

5. **Hooks e deploy com Git:**
    - Hooks: executam algum código quando determinado evento acontece, através de um arquivo Shell Script, onde o nome do arquivo representa o evento. Encontram-se dentro da pasta .git/hooks;
    - Para fazer deploy com o git: ir no hooks, entrar no post-receive e colocar **git --git-dir={caminho_da_pasta_do_servidor} --work-tree={caminho_da_pasta_web} checkout –f;**