Relatório 9 - Git e Github para Iniciantes

Edryck Freitas Nascimento

1. Introdução

O objetivo desta aula foi aprender sobre Git, GitHub e GitHub Desktop. A tarefa consistia em assistir um minicurso da Udemy (Git e Github para Iniciantes) e um vídeo do Youtube (Como usar o Github Desktop).

2. Desenvolvimento

Algo bem importante para iniciar é saber o que é o Git, o Github e o Github Desktop.

Git: De forma bem simplificada, é uma ferramenta para versionamento.

GitHub: Muitos confundem achando que é o mesmo que o Git, mas é uma plataforma para armazenamento de código na nuvem, em outras palavras: um repositório remoto.

GitHub Desktop: Para quem não gosta de ficar no terminal escrevendo comandos, é algo como uma UI para o terminal. Ela tem todas as funções do terminal, ou se não a maioria das funcionalidades.

Se conectarmos elas, seria algo assim:

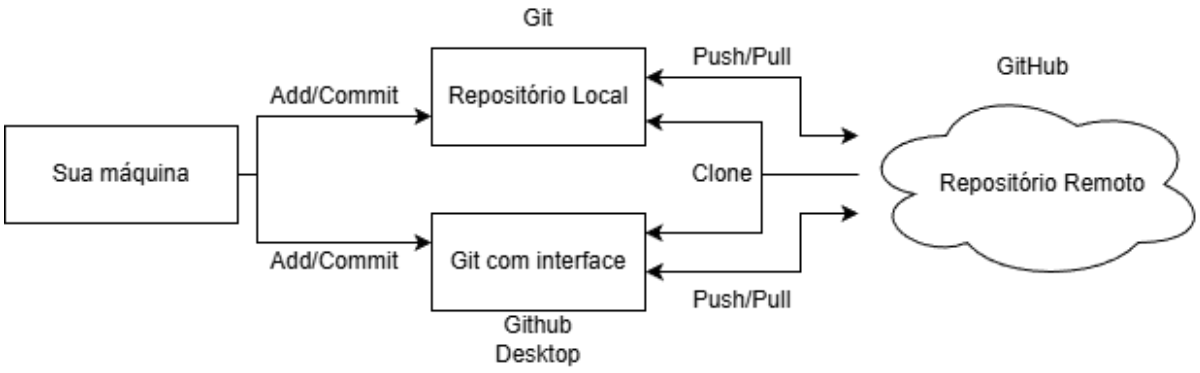

Figura 1. Git, GitHub e GitHub Desktop (autoria própria).

Você realiza o versionamento do código com o Git e faz um backup dele no repositório remoto no GitHub. Os mesmos comandos do Git também são disponíveis no GitHub Desktop, basicamente a diferença entre eles é mais que um é em terminal e outro tem uma GUI.

A configuração inicial e essencial do Git é a identificação do autor, nome e email, uma vez que configurado, o ciclo de vida de um arquivo no Git passa por quatro estados iniciais:
***Untracked*:** O Git não sabe da existência desse arquivo.

***Unmodified*:** O arquivo existe, mas não sofreu nenhuma alteração.

***Modified*:** O arquivo foi alterado. (Usamos o comando *add* para colocá-lo no estado *staged*)

***Staged*:** Arquivo pronto para ser realizado o *commit*. (Após utilizar o comando *commit* ele vai para o estado *unmodified*)

Como dito anteriormente, o Git cuida das versões na máquina local, a importância do GitHub é que após colocar o projeto em um repositório remoto, isso vai permitir o trabalho em equipe, assim várias pessoas vão conseguir trabalhar juntas em um mesmo projeto. Para realizar essa conexão do local com o remoto, é usado as chaves SSH, ela que vai garantir uma comunicação segura entre a máquina e o repositório remoto, isso sem precisar ficar digitando senhas constantemente.

Um dos conceitos mais importantes, acredito eu, são as ramificações (*branches*), as *Branch* permitem criar uma funcionalidade ou corrigir um erro sem mexer no código principal. Ao trabalhar em uma *Branch* para adicionar as alterações ao código principal usamos dois comandos:

Merge: É o processo de “fundir” o trabalho feito no código principal.

Rebase: É uma forma alternativa ao primeiro que organiza o histórico de *commits* de forma linear e mais limpa.

Entretanto, cada um tem suas particularidades, a escolha de uso dos dois depende do contexto na qual irá ser utilizado, o *merge* é uma operação não destrutiva, mas irá criar um *commit* extra onde vai juntar as duas *branches* e deixa um histórico poluído, já o *rebase* não adiciona um *extra* e mantém um histórico linear, porém, você irá perder a ordem cronológica dos *commits*.

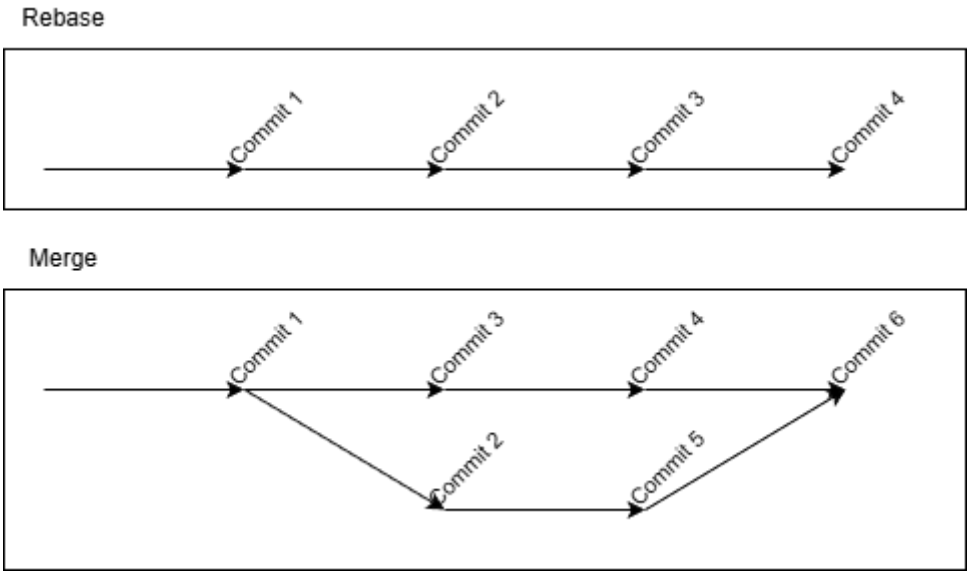

Figura 2. *Merge* vs. *Rebase* (autoria própria).

3. Conclusão

Na atualidade, saber usar essas ferramentas é algo que todo desenvolvedor deveria saber. O Git te dá o controle total sobre o histórico do projeto, assim permitindo ver todo o histórico de mudanças do projeto, corrigir erros e adicionar novas funcionalidades sem mexer no código principal, já o GitHub atua como um espaço de backup seguro para o código e como currículo para mostrar os projetos que você já desenvolveu e participou. O GitHub Desktop é uma excelente porta de entrada para quem prefere uma visualização mais clara e menos dependente de comandos memorizados.

Referências:

Curso **Git e Github para iniciantes**:

https://www.udemy.com/course/git-e-github-para-iniciantes/

Vídeo **Como usar o Github Desktop**:

https://www.youtube.com/watch?v=Fj3gtbaF8WA