**Como usar o Git e Github na prática**

**Início do vídeo**

Oie gente sejam bem vindos a mais um vídeo aqui do canal, eu sou a Rafaella Ballerini e hoje eu vou mostrar pra vocês como utilizamos o GIT na prática

No [vídeo anterior](https://www.youtube.com/watch?v=DqTITcMq68k) eu expliquei os conceitos de alguns termos técnicos que utilizamos quando estamos usando o git, então [nesse vídeo](https://www.youtube.com/watch?v=UBAX-13g8OM) vou mostrar na prática como isso acontece.

**Instalando o GIT**

* [Link com os downloads](https://git-scm.com/downloads)

**Criar um projeto novo**

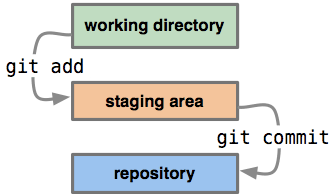
* Criar uma nova pasta no PC pra isso chamada Git Tutorial
* Abrir o VSCode nessa pasta
* Criar um novo arquivo README.md
* Escrever dentro dele Olá, nesse projeto você aprenderá alguns comandos do Git
* Salva o arquivo

Agora então é hora de usarmos o Git

* Abre o Git Bash que foi instalado na máquina (pode ser pelo terminal do VSCode mesmo)
* git init para inicializar o repositório

Foi criada uma pastinha .git e é ali que toda a mágica acontece, então não apague

* git add README.md para colocar o arquivo na área de stagging

[](https://camo.githubusercontent.com/6c19eae2fb7be4ce9a9e054164e6f60f2d2af38cef5ad13460ec15e65029bb15/68747470733a2f2f69312e77702e636f6d2f7777772e6d61726b75732d676174746f6c2e6e616d652f6d6973632f6d6d2f73692f636f6e74656e742f6769745f6769745f6164642e706e67)

Esse add é necessário antes de darmos o commit de fato, mas por que isso? No final do vídeo explico para vocês

* git commit -m "primeiro commit" para de fato dar o commit no repositório
* git branch -M "main" para alterar o nome da branch principal de master para main (isso é uma boa prática atualmente recomendada)

**Interfaces Git**

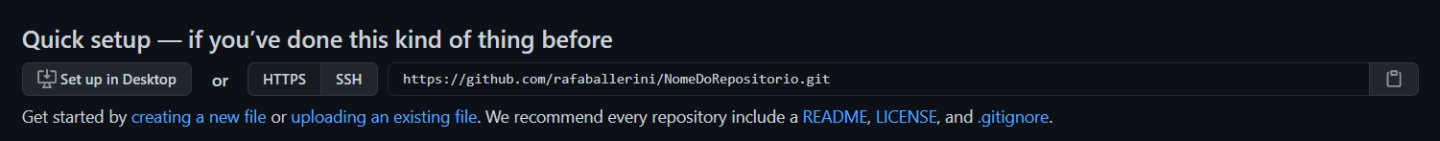
Beleza, recebemos a confirmação de que o commit aconteceu, mas isso tá um pouco abstrato ainda né? Existem algumas [interfaces legais do git](https://git-scm.com/downloads/guis) que você pode fazer o download para poder visualizar de fato como está o projeto, o que foi alterado em cada commit, quando foi alterado etc. Aqui eu vou mostrar pra vocês a usarem direto no Github.

**Repositório no Github**

* Depois de você ter criado a sua conta na plataforma, você irá em Criar novo repositório

Você vai preencher com as informações do projeto, então dar o nome do repositório, colocar uma breve descrição e criar

Logo depois vai aparecer essa página um pouco cinza e confusa e com vários comandos (pode até perceber que alguns deles jpa usamos), mas o que você tem que fazer é bem simples, apenas copie o link que aparecer para você

[](https://camo.githubusercontent.com/beb6fcc87770c8254663452798a95344fadda570b4a7bc0acf1e692e82933a2b/68747470733a2f2f6d656469612e646973636f72646170702e6e65742f6174746163686d656e74732f3833313937343135323636373339383231342f3833363832383930353835393138363730382f756e6b6e6f776e2e706e673f77696474683d31343430266865696768743d313431)

Lembra do conceito de remote que eu expliquei pra vocês no último vídeo? Nós iremos utilizá-lo agora

* Para passar o commit do meu repositório local (da minha máquina) para um repositório na plataforma do Github, usamos o git remote add origin <link do repositório>
* origin é o nome utilizado para referenciar o nosso repositório

Agora já temos o nosso repositório local conectado com o respositório do Github, porém o commit que damos na máquina não sobe automaticamente para a plataforma

* Para isso precisaremos empurrar, enviar para lá com o git push -u origin main

Agora se recarregarmos a página iremos ver o nosso arquivo aqui na plataforma!

**Alterando e adicionando arquivo**

Beleza, agora que temos o nosso repositório no Github configurado direitinho, podemos usar e abusar do que o Git oferece, afinal é pra isso que estamos utilizando ele né? Primeira coisa que faremos então é alterar esse arquivo que já commitamos

* Adiciona mais uma frase no arquivo Essa é uma alteração
* Além disso iremos criar um novo arquivo Projeto.md, onde escreveremos Esse é o arquivo onde desenvolverei o meu projeto
* Agora então precisamos subir essa alteração, pra isso seguiremos os mesmos passos de git add . (agora ponto . pois adiciona todos os arquivos) e git commit -m "Primeira alteração"
* Lembrando que para alterar algo no nosso respositório do Github precisamos dar o push, então git push origin main (sem o -u)

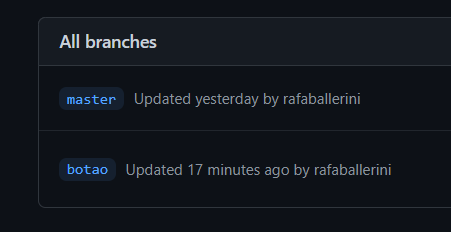
Se olharmos agora o nosso código no Github, ele terá sido alterado, e não só isso, se clicarmos no nome do commit, podemos ver exatamente as alterações que foram feitas nele. O verde com + e o vermelho com - mostra, os conteúdos que foram adicionados e editados dentro do código. Aqui nesse botão poderemos ver todos os commits já feitos anteriormente, então se clicarmos em algum deles, veremos exatamente o que havia sido alterado, além de claro, vermos o código como era. Incrível né?

**Branch**

Até agora tudo o que fizemos de alterações e mandamos de commit, foi na nossa main, que é aquela linha do tempo principal. Agora vou mostrar pra vocês como criamos uma branch e depois como que juntamos ela com o código que já está na main (lembrando que ela é uma linha cronológica adicional/alternativa a principal) E outra, a branch pode ser criada tanto para quando você for fazer uma alteração em um arquivo, quando para adicionar outro arquivo dentro do projeto ou mesmo excluir.  
Obs. *Lembrem que eu estou aqui editando um arquivo markdown, porém isso tudo vale para qualquer tipo de arquivo com qualquer extensão*

* Nesse caso vamos adicionar um novo arquivo para desenvolver a nossa feature Botão
* Então a primeira coisa que fazemos é git checkout -b "novo-botao", assim criando uma branch para ele Esse comando além de criar a branch já entra nela com o checkout, inclusive se olharmos agora aqui no meu VSCode, estamos dentro dela.
* Vou então criar o arquivo, criar o botão.md "aqui eu crio o botão"
* E agora fazemos o passo a passo que já sabemos, colocamos a nossa alteração em stagging com o git add . e commitamos com o git commit -m "novo botão"
* Para enviarmos agora que vai ser diferente. Vocês lembram que utilizávamos o git push orgin main né? Porém main era aquela branch principal. Agora então usaremos git push origin botao

Agora se olharmos o nosso Github, veremos que tem 2 branches, a main e a botao

[](https://camo.githubusercontent.com/60d78c06a07fb7f2c13c2dbe116a1162b20b4ad5401746f7041ebbf34bf15fd2/68747470733a2f2f6d656469612e646973636f72646170702e6e65742f6174746163686d656e74732f3831323331333734323139323237393631322f3833363832303637303033373632323835342f756e6b6e6f776e2e706e67)

Vamos supor que eu ainda não tivesse terminado de desenvolver o botão, eu poderia continuar tranquilamente na branch botao até terminar!

Mas Rafa, e se eu precisasse por algum motivo voltar naquela branch main e desenvolver a partir do que deixei lá? Sem problemas, a única coisa que você precisa fazer nesse caso é git checkout main, e pra voltar depois é só git checkout botao novamente

Beleza! Agora desenvolvi tudo o que queria aqui na branch botao, como que junto ela com a main sem problemas?

**Merge**

* Agora o que precisamos fazer é ir para a nossa branch principal git checkout main e lá faremos o merge com a branch botao que criamos, com git merge botao

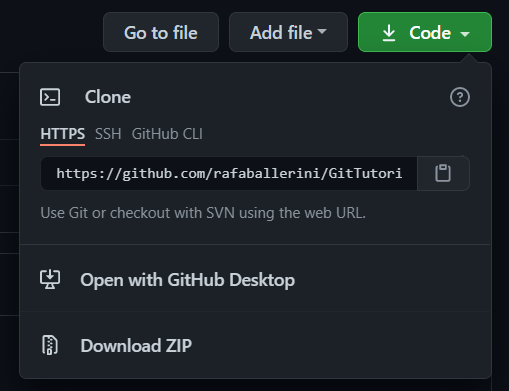
Pronto, agora tudo o que tinha de alteração na branch botao juntou com a main

* Para finalizar então, vamos jogar lá no Github isso tudo com o git push origin main

**Clone**

Como vocês podem baixar meu código?

Sempre que você entrar em um repositório, seja o seu ou o de qualquer outra pessoa, terá esse botão Code, que quando você clica aparece um link:

[](https://camo.githubusercontent.com/b6c163daa5b76b8e9dcede4dab60c5b3f573bbdcbde2113572800b24428c2173/68747470733a2f2f6d656469612e646973636f72646170702e6e65742f6174746163686d656e74732f3831323331333734323139323237393631322f3833363832333536343531333730353939342f756e6b6e6f776e2e706e67)

* Você irá copiar esse link e levar ele lá pro nosso terminal
* O comando para puxar o projeto para a sua máquina é o git clone https://github.com/rafaballerini/GitTutorial.git

Não é necessário criar um repositório antes disso, como fizemos anteriormente com o git init. Dessa vez, basta abrir o terminal e clonar o projeto e tudo aparecerá!

**Pull**

E se eu fizer uma alteração no repositório, como vocês podem atualizar na máquina de vocês?

* Basta vocês executarem o comando git pull, ele irá puxar todas as alterações feitas no repositório do Github para o seu repositório local

**Fork**

Mas Rafa quando eu fiz o clone do seu repositório ele não apareceu no meu Github. Existe a ferramenta fork, que é bem mais simples para fazer isso Você só precisa apertar nesse botão dentro do repositório e TCHANAM! Ele aparece automaticamente lá na sua conta:

[](https://camo.githubusercontent.com/1df0db71834b8c3dbb87321ce0c7d48a8125d13651531f0b1156101e0e7470eb/68747470733a2f2f6d656469612e646973636f72646170702e6e65742f6174746163686d656e74732f3833313937343135323636373339383231342f3833363832363638373633343430373433342f756e6b6e6f776e2e706e67)

**Pull request**

O último conceito que quero ensinar para vocês é o de Pull Request, vamos entender como ele funciona:

* Após você ter dado um fork no projeto e ele ter ido pra sua conta, você poderá alterar o projeto e adicionar as funcionalidades que deseja
* Você pode por exemplo dar um fork no meu repositório de Formulário para adicionar uma validação de campos ou qualquer outra coisa que acha válido
* Depois disso, você poderá salvar o projeto, dar o git add ., git commit -m "validação de botões" e git push origin main

Quando você for olhar o seu Github, verá que existe uma mensagem parecida com a seguinte:

[](https://camo.githubusercontent.com/b13361be0f001ff1166f65dee23ded79a869768e2758702f75b1e83878e2258e/68747470733a2f2f6d656469612e646973636f72646170702e6e65742f6174746163686d656e74732f3833313937343135323636373339383231342f3833383939303938333835323435383033352f756e6b6e6f776e2e706e67)

Isso significa que a branch do seu repositório está 1 commit "na frente" da branch original

O que você deve perceber agora é esse botão que aparece em seguida:

Ele servirá para caso você deseje enviar para o dono do repositório original uma solicitação de pull, ou seja, fazer com que ele puxe as alterações que você fez no seu repositório para o repositório dele, original

Ao clicar nesse botão, você será direcionado para uma página que fará a avaliação se esse pull request terá conflitos ou não com o código no repositório original. Caso não tenha, bastão clicar no botão de Create pull request

Você irá colocar um nome intuitivo, que demonstre a funcionalidade adicionada e o ideal é que você também crie uma boa descrição do que desenvolveu, não somente explicando o que é, mas ensinando ao dono do repositório original a forma como ele poderá testar também

Depois disso, basta esperar para que o dono da branch original aceite o seu pull request

**Finalização**

Existem diversas outras funcionalidades do Git e do Github, porém tenho certeza que com tudo isso que vocês viram hoje vocês já conseguem desenvolver um projeto de uma forma bem legal

Recomendo sempre vocês darem uma olhada na [documentação do Git](https://git-scm.com/doc), pois qualquer dúvida que apareça pode ser respondida por lá na explicação

**Não esqueçam de deixar o like e se inscrever no**[**canal do Youtube**](https://youtube.com/RafaellaBallerini)**❤**

Até semana que vem, um beijo!