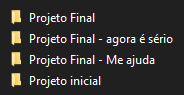
**Para saber mais: Para saber mais: Git e Github**

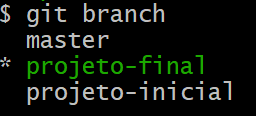
Caso você não tenha conhecimento sobre git, não se preocupe! Neste curso não utilizaremos com frequência , mas para a sua carreira é muito importante entender como utilizar o Git e Github.

## Git

Git é a ferramenta utilizada pelos desenvolvedores para controlar as versões do seu código, ou seja, ao invés de você ter algo assim:



Você pode ter algo assim:



Ao invés de você ter pastas diferentes com códigos parecidos, você terá branches com códigos parecidos!

Branches seriam uma representação dessas pastas que você criaria sem o Git, e caso você queira ir para uma destas branches, você pode utilizar o comando git checkout, por exemplo:

* git checkout projeto-inicial < estou na branch projeto-inicial
* git checkout projeto-final < estou na branch projeto-final
* git branch projeto-final-2 < criei uma nova branch projeto-final-2
* git checkout projeto-final-2 < entrei na branch projeto-final-2 que eu acabei de criar
* git branch < sem especificar nenhuma branch, o git me mostra todas as branches que eu tenho localmente

Viu? Git é algo muito útil e com o tempo fica muito intuitivo!

Beleza, eu já sei como criar uma branch e como andar de uma branch para outra, mas como eu faço para criar ou pegar um projeto com git???

Então, para criar um projeto com git, primeiro você precisa saber se o projeto já contém o git, para fazer isto, tem 2 formas principais:

1. Olhando se na raiz do projeto existe uma pasta .git;
2. No terminal, escreva qualquer comando git (como o git branch, por exemplo) e veja se o terminal entende, se ele entender, o seu projeto contém git, se não, provavelmente aparecerá algo com not a git repository.

E para pegar um projeto que já tem git??

Bom, para pegar um projeto que tem git, primeiro você precisa saber onde achar este projeto, certo? É aí que entra o Github!

## Github

O Github nada mais é do que um site que guarda projetos git! Estes projetos são chamados de repositórios, e você pode ter repositórios públicos ou privados.Todos os repositórios que você tem acesso podem ser **clonados** ou **“forkados”**. E o que significa clone ou fork?

Antes de comentar sobre estes dois termos, tenho que explicar sobre algo chamado remote. Todo repositório pode ter um ou mais remotes, que nada mais são que link de repositórios online (como no Github) que você pode trocar informações, sendo pegar código novo ou enviar um código novo que está na sua máquina. O nome padrão do remote normalmente é origin, mas você pode adicionar o nome que você quiser em outros remotes, caso você queira fazer esta conexão.

### clone

Com o clone, você copia o código para a sua máquina, porém o remote permanece o da pessoa que você clonou, ou seja, se você copiar o código do React por exemplo, o remote é o link do código real, do React. Como você não tem permissão para enviar código para o repositório oficial do React, fazer isto seria negado.

### fork

Aí que entra o fork! O fork faz uma cópia daquele repositório no seu Github! Ele não copia o código para a sua máquina, mas após esta cópia, você poderá fazer um clone do seu repositório (o repositório copiado via fork) e agora o remote estará apontando para o seu Github, não mais para o Github oficial!

Agora que você sabe mais sobre estes termos, fique a vontade para fazer um clone ou um fork do [repositório do projeto!](https://github.com/lfrprazeres/organo.git) ou se continuar com dúvidas, indicamos [o curso de Git e Github aqui da Alura](https://cursos.alura.com.br/course/git-github-repositorio-commit-versoes).