GIT E GITHUB: Entenda de uma vez por todas o que é e como funciona essas duas ferramentas.

Muita gente, mas muita gente mesmo, se pega confundindo os termos **GIT** e **GITHUB**, mas fique tranquilos que vamos explicar esses conceitos de uma vez por todas, tenho certeza que após você ler esse artigo conseguirá distinguir as duas ferramentas e utilizá-las da melhor forma possível.

GIT

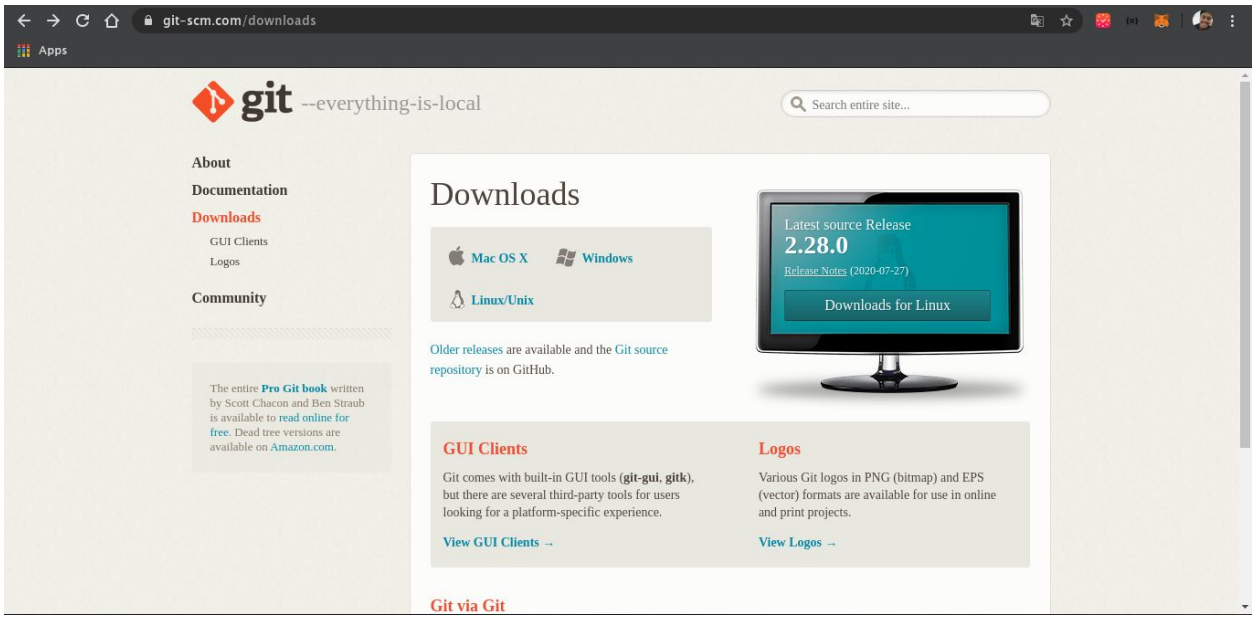
CONCEITO

O GIT é um sistema de controle de versão de código desenvolvido tanto para lidar com projetos pequenos, quanto projetos maiores, mas para você realmente entender o que é o GIT vamos ao nosso exemplo: imagine que você faz parte de um time de desenvolvimento, cuja função é desenvolver um sistema de gerenciamento. Em uma equipe de desenvolvimento temos diversos tipos de profissionais que desempenham funções distintas, quando estamos trabalhando com código cada programador realiza modificações em certas partes do código fonte do sistema, logo como fazemos para juntar cada modificação que cada desenvolvedor realizou no código? Como modificamos uma parte do sistema sem alterar a versão em produção? Pegamos um pen drive e colocamos o código e passamos para o colega do lado? NÃO... é aqui que chegamos ao conceito de versionamento de código, essa ferramenta permite que o desenvolvimento seja como uma linha no tempo, onde temos linhas do tempo (branch) e diversos pontos na história (modificações, commits) , sendo a linha do tempo principal (master) o projeto na versão final e as outras linhas do tempo paralelas são modificações que os desenvolvedores do time realizaram. O GIT ajuda você a ter mais controle do projeto, permite que visualize modificações específicas em um determinado ponto na história, permite que você volte em um ponto específico e entre outras funcionalidades, agregando não somente para o desenvolvimento, mas também para o time.

INSTALANDO O GIT NO SEU COMPUTADOR

Como esse é um artigo que tem o objetivo de levar você a entender o GIT e GITHUB, vamos ao uma abordagem mais prática.

1. Entre nesse link https://git-scm.com/downloads e baixe a versão para o seu sistema operacional.
2. Depois de baixado e instalado verifique com o comando “**git --version**” no seu terminal(cmd ou terminal).
3. Caso apareça a versão do GIT, ele foi instalado com sucesso na sua máquina.



INICIALIZANDO UM REPOSITÓRIO LOCAL COM O GIT

Para que o GIT consiga monitorar as alterações do nosso projeto precisamos primeiramente inicializar um repositório GIT no local do nosso projeto. OBS: “repositório” é um diretório qualquer, ou seja uma pasta onde o GIT mantém todos os arquivos necessários para o controle de versão.

Para inicializar um repositório local com o GIT:

1. Entre na pasta onde o projeto se encontra e na raiz da pasta do projeto abra o terminal.
2. Digite o comando “**git init**” no terminal para inicializar um repositório local no diretório do seu projeto.
3. Se tudo der certo o GIT retornará uma mensagem “**Initialized empty Git repository in e “aqui caminho completo do diretório”**. Agora o GIT iniciou um repositório onde vai conseguir manter todo o versionamento do nosso projeto.

VERIFICANDO O STATUS ATUAL DO PROJETO COM O GIT

Na ferramenta GIT existe um comando que permite que ele verifique, em todo o seu projeto, se existe alguma modificação ou se algum novo arquivo foi adicionado ou deletado. É muito importante pois você consegue visualmente ter o controle do projeto. Para realizar esse procedimento utilize o comando "**git status**" no terminal. Onde a ferramenta te mostra na branch onde você se encontra e quais são os arquivos a receber o commit (modificados, adicionados ao staged).

PERMITINDO QUE O GIT MONITORE AS MODIFICAÇÕES DO PROJETO

Essa é uma das principais etapas, pois através dela que o GIT vai conseguir observar sempre que um arquivo ou funcionalidade é modificada no nosso projeto, dizendo para nós quais arquivos e onde eles foram modificados. O GIT só consegue observar as modificações depois que elas são adicionadas a um estado chamado de **staged**, basicamente o git registra o estado atual de todo o conteúdo do projeto e verifica se existem modificações baseado no último commit, se existe modificações nós precisamos adicioná-las ao staged, uma área que prepara todas essas modificações para o commit, com o comando "**git add nome-do-arquivo**" adicionando um arquivo específico ou "**git add** ". Para adicionar todas as modificações de uma só vez, o GIT assim que esses arquivos são adicionados registra o novo estado e consegue verificar se existe modificações ou não no projeto.

REALIZANDO COMMITS DAS ALTERAÇÕES

O próximo passo depois de ter adicionado os arquivos criados ou modificados ao staged é fazer o commit. Imagine o commit como se fosse uma caixa, onde o GIT guarda todas aquelas alterações que foram feitas e lacra essa caixa e adiciona como um ponto na história, já que estamos falando de linha do tempo, o ponto interessante é que o GIT te permite navegar entre esses pontos (commits) na sua linha do tempo (branches), o GIT te dá o super poder de voltar no tempo, e esse é o ponto central do sistema de versionamento. Fez alguma alteração, entretanto precisa voltar a versão anterior? O GIT permite que faça isso, pois temos a capacidade de adicionar pontos na história (commits).
Para realizar os commits das modificações basta utilizar o comando "**git commit -m "mensagem explicando o commit**"" no terminal. OBS: a flag "**-m**" do lado da palavra commit significa mensagem, é a mensagem que você vai passar explicando e detalhando aquele commit.

CRIANDO LINHAS DO TEMPO PARALELAS

Como foi dito anteriormente o GIT funciona como se fosse um linha do tempo, onde a principal é a master. OBS: quando falamos de linha do tempo estamos falando de branches ou ramos.
O GIT permite criar quantas linhas do tempo forem necessárias, então vamos partir para mais um exemplo para você entender melhor. Lembra do exemplo acima, do time de desenvolvimento? Então, como foi dito cada programador trabalha em uma parte específica do código fonte, quando um desenvolvedor tem que realizar um teste ou uma implementação de uma funcionalidade ele cria um branch, ou como estamos falando neste artigo uma linha do tempo paralela, onde essa linha do tempo herda todas as características da linha do tempo principal e a partir dela podem ser feitas modificações que só irão ter efeitos naquela linha do tempo (branch).
O principal benefício é que você consegue modificar, deletar e inserir sem alterar a versão atual em produção (linha do tempo principal) e também sem ter qualquer contato com as modificações que seus colegas de time estão fazendo naquele momento.
O comando para essa funcionalidade é o “git branch nome-da-branch” para criar uma nova branch e caso você queira criar uma nova branch e já se redirecionar para ela o comando é “**git checkout -b nome-da-branch**”, caso queira saber em qual branch você se encontra basta digitar o comando “git branch” que o git lista as branches disponíveis e mostra em destaque a que você se localiza. Vale ressaltar que, também temos o comando “**git switch nome-da-branch**” para ir de uma branch para outra, o GIT permite que você realize a função de diversas formas, todavia para facilitar a compreensão vamos nos ater aos principais comandos.

JUNTANDO ALTERAÇÕES DA BRANCH DE DESENVOLVIMENTO COM A PRINCIPAL

Um das funcionalidades mais interessantes do GIT é a possibilidade de juntar a linha do tempo paralela com a principal, ou seja, todas aquelas alterações que o desenvolvedor fez na branch de desenvolvimento, agora podem ser mescladas com o ramo principal. A tarefa é relativamente simples, basta você ir para a branch na qual quer que as alterações sejam mescladas e realizar o **merge** das alterações da outra branch. O comando próprio para essa tarefa é o “**git merge nome-da-branch**”.

GITHUB

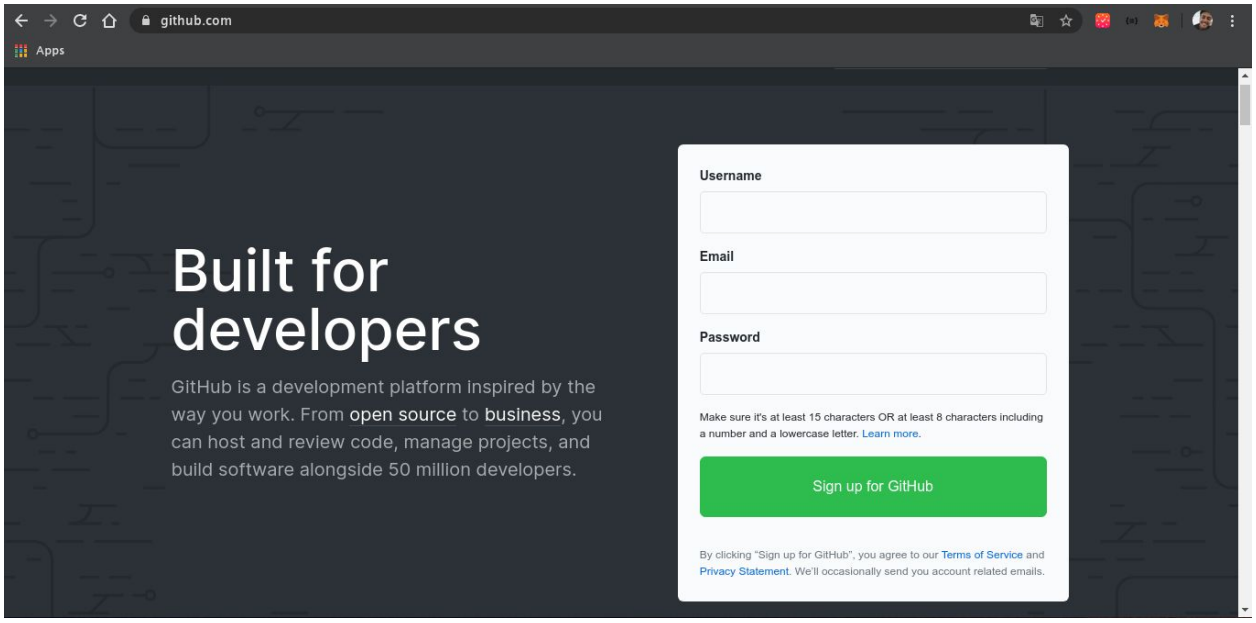
CONCEITO

O GITHUB é uma plataforma que serve para a hospedagem de código-fonte, compartilhamento de informações, gerenciamento de projetos e usa o sistema de controle de versão GIT. Ele permite que desenvolvedores ao redor do mundo consigam contribuir tanto para projetos open-source, quanto para projetos privados.

CADASTRO NA PLATAFORMA

Para eu utilizar tenho que ser desenvolvedor? NÃO...qualquer pessoa do mundo pode criar uma conta gratuita e utilizar a ferramenta. É tão simples como qualquer outro cadastro, basta entrar nesse site: https://github.com/ e nos campos de cadastro digitar o nome de usuário, email e senha. Logo após confirme o email e pronto! Já pode fazer login e usar tudo o que o github pode oferecer.

Segue a imagem abaixo:

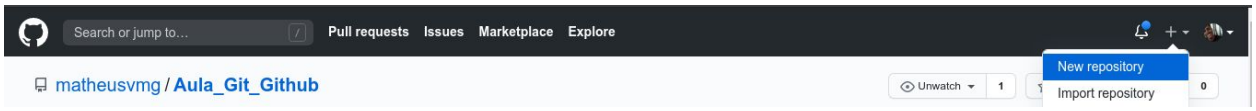


CRIANDO UM REPOSITÓRIO NO GITHUB

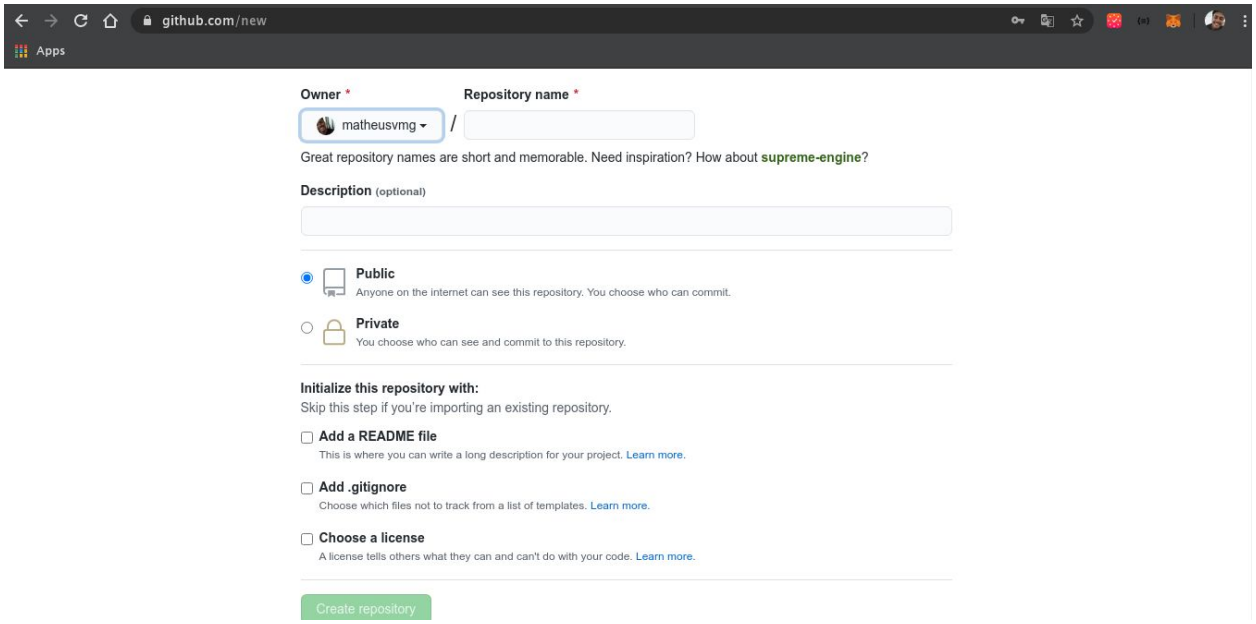
1. Faça login na sua conta do github

2. Clique no botão de + no canto superior direito
3. Clique no ítem de menu novo repositório
4. Preencha o nome do repositório
5. A descrição do repositório é opcional, caso queira colocar
6. Marque se esse repositório será público ou privado
7. Se você for importar um repositório local não precisa marcar nenhuma das opções abaixo

Passo 2 e 3



Passo 4 em diante



SUBINDO O PROJETO PARA O REPOSITÓRIO REMOTO

Aqui estamos na etapa final. Quando o time de desenvolvimento já fez todas as alterações no projeto, validaram e juntaram todas as modificações, e agora falta enviar o projeto para o repositório remoto (servidor).
Vale ressaltar que existem diversas opções de ferramentas e serviços para a hospedagem desse projeto, nesse artigo abordaremos o github.
Existe duas maneiras de enviarmos os arquivos do nosso projeto para o servidor do github, a primeira é através da interface do github adicionando os arquivos arrastando para a interface ou abrindo pelo explorador de arquivos. A segunda é como iremos prosseguir nesse artigo, através da linha de comando.

1. Verifique o status do seu projeto com o comando "**git status"**.
2. Se tem arquivos para serem adicionados, adicione-os com o "**git add nome-do-arquivo**" ou "**git add**", para adicionar tudo.
3. Comente os arquivos adicionados pelo comando "**git commit -m "mensagem explicando o commit aqui**".
4. Aponte para qual repositório remoto que você deseja enviar os arquivos do projeto com o comando "**git remote add origin url-do-repositório**".
5. Aponte quais branches você deseja enviar para o repositório remoto, no nosso caso só a master, pois é a linha do tempo principal, com o comando "**git branch -M master**".
6. Envie tudo para o servidor com o comando "**git push -u origin master**".
7. Configure o seu nome de usuário com o comando "**git config --global user.name "nome-do-usuário**"".
8. Configure o seu email com o comando "**git config --global user.email "email-cadastrado-no-github**"".
9. Caso peça, digite seu nome de usuário/email e senha e aperte enter.

Pronto! Agora é só atualizar a página lá no repositório do github que os arquivos do projeto estarão lá.
O GIT e o GITHUB são ferramentas fantásticas que se usadas em conjunto conseguem potencializar e muito o processo de desenvolvimento.

Gostou do artigo? Não esqueça de deixar seu feedback lá na nossa página do instagram @_technoday e compartilhar o artigo.