CURSO GIT et GITHUB

1. **O que é GIT e GITHUB**

GIT – é um sistema de controle de versões distribuído, usado principalmente no desenvolvimento de software, mas pode ser usado para registrar o histórico de edições de qualquer tipo de arquivo.

GITHUB – é uma plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git.

1. **Como Inicializar um repositório**

* Abrir uma pasta
* Inicializar o nosso repositório – clicar com o botão direito do mouse dentro da pasta e depois clicar no comando > git bash here
* Abrimos a linha de comando do próprio GIT
* Após digitar o comando > git init
* Criar um arquivo de qualquer extensão
* Para verificar o status do GIT se está atualizado digitamos o comando > git status

1. **Como adicionar arquivos no controle de versão GIT**

* Para adicionar o arquivo você digita o comando > git add “meu código.txt”
* Verificando novamente com o comando > > git status observamos que o arquivo foi adicionado.
* Caso você tenha vários arquivos para adicionar você utiliza o comando > git add .

1. Como criar versões do código com COMMIT

* Para criar a primeira versão vamos utilizar o comando > git commit -m “commit inicial” (você digita uma mensagem entre parênteses informando o que foi feito naquela versão
* Se você estiver rodando este comando pela primeira vez, vamos utilizar o comando > git config --global user.email ”digite o seu email” e outro comando > git config --global user.name “digite o seu nome” para configurar o seu email e o usuário no GIT.

1. Como enviar alterações usando GIT PUSH

* Para enviar a nossa primeira versão do código para o nosso servidor nas nuvens, digite o comando > git push
* Após esse comando, ele não sabe para onde enviar essas alterações, então teremos que definir para onde deveremos enviar essas informações criando um link do nosso projeto na nuvem no GITHUB
* Vamos entrar no IGTHUB > clicar no sinal de + e selecionar o new repository
* Colocar o nome do repository e a descrição do projeto (opcional)
* Sempre adicionar o arquivo README.FILE
* Para trazer o nosso arquivo local para o GITHUB, devemos copiar o link acima e na linha de comando vamos digitar o comando > git remote add origin https://github.com/JGlaucioSantos/projeto2---curso-GIT-e-GITHUB (colando o seu link)
* Com isso, acabamos de definir para onde o meu código deve ser enviado no GITHUB.
* Agora para enviar de fato o nosso código, devemos digitar o comando > git push
* Ele não vai funcionar porque ele não sabe para qual BRANCH você deverá enviar o seu código. Com isso vamos digitar o seguinte comando > git push –set-upstream origin master

1. Ciclo de atualização de códigos

* Quando o arquivo teste.txt foi modificado, ou dos seus códigos você deve dar um comando > gif status
* Após isso você terá essa informação > modified: teste.txt
* Você deve digitar o comando > git add .
* Após isso nos vamos fazer um novo commit da nova versão utilizando o comando > git commit "alterações no texto"
* Após criado o commit eu digito o comando > git push para enviar a alteração feita.

1. Como verificar históricos de atualizações

* Para saber o histórico das versões você deve digitar o comando > git reflog
* Observando sempre que o a versão mais nova fica sempre em cima

1. Como navegar entre versões do seu código

* Quando você encontrar um erro na versão anterior, você devera digitar o seguinte comando > git reset –hard (após digita o ID da versão que você quer alterar) no caso eu vou digitar o id da primeira versão > d5908dd
* E para voltar a versão mais atualizada, devemos utilizar o comando > git reflog e após digitar o comando > git reset –hard (após digita o ID da versão que você quer alterar) no caso eu vou digitar o id da primeira versão > d5908dd

1. O que são Branches e como criar uma

* ***O que é um branch*** - Uma ramificação no git é um ponteiro para as alterações feitas nos arquivos do projeto. É útil em situações nas quais você deseja adicionar um novo recurso ou corrigir um erro, gerando uma nova ramificação garantindo que o código instável não seja mesclado nos arquivos do projeto principal. Depois de concluir a atualização dos códigos da ramificação, você pode mesclar a ramificação com a principal, geralmente chamada de master.
* Para saber o número de branch você digita o comando > git branch
* Para você criar uma branch você digita o comando > git branch (após coloca o novo nome da nova branch que vai ser criada) staging
* Após nos sabermos quais são as branches que existem, temos que verificar qual a que está funcionando no momento. Para saber, o branch se destaca pelo sinal de asterisco com a com verde **>**

**\* master**

* Para mudar para o branch que queremos trabalhar, vamos digitar o comando > git checkout (colocar o nome do branch) staging
* Após esse comando, ele vai passar para a branch que iremos trabalhar ou seja, vai ficar com o asterisco e na cor verde > \*sataging
* Com isso vamos fazer as modificações necessárias, salvar e novamente dar um comando >git status. Nesse caso vamos verificar que existe uma nova alteração a ser feita no branch staging.
* Vamos adicionar as alterações dando um comando > git add .
* Agora vamos fazer um novo commit > git commit -m “alteração do texto”
* Agora vamos verificar como está com o comando > git status
* Agora vamos enviar as alterações para o GitHub com o comando > git push
* Vamos receber um comando para efetuar o envio > git push --set-upstream origin staging copiamos fazemos o comando e enter

1. Git merge como unir o código

* Agora nós vamos fazer a atualização do branch master unindo as alterações do branch staging com o branch master.
* Primeiro vamos entrar no branch master com o comando > git checkout master
* Agora vamos verificar qual branch vamos trabalhar colocando o comando > git status
* Ok agora estamos no branch \*master
* Após vamos puxar as alterações do branch staging para o branch master com o seguinte comando > git merge staging
* Agora vamos dar um comando > git push e vamos verificar que no GitHub a branch master está atualizada
* Atenção!!! Antes de voce fazer o merge, nós temos que puxar todas as atualizações que estão no meu servidor para a minha máquina para ter certeza de que eu estou unindo os códigos nas versões mais atualizadas possíveis
* entre na sua branch principal e digite o comando > git pull
* Ai sim, após isso vamos atualizar a versão master usando o comando > git merge staging
* Resumo:
* Git pull da branch principal
* Gerar uma nova branch a partir da branch principal
* Trabalhar e adicionar novas funcionalidades na nova branch que criou
* Finalizar o trabalho na branch temporária
* Git checkout na branch principal
* Git pull
* Mergiar (unir) o código da branch temporária com a branch principal (depois de testar)
* Git push da branch principal

***EXEMPLO DE UM TESTE – Você recebeu a tarefa de criar um sistema de login***

1. Passo

Digite o comando git branch – para identificar qual branch você está trabalhando

1. Passo

Digite o comando > git pull – atualizar o código

1. Passo

Digite o comando > git checkout -b (nome da branch) sistema-de-login master - criar uma nova branch com base na branch principal em um só comando

1. Passo

Confirmar a criação de uma nova branch digite o comando > git branch

1. Passo

Adicionar o código ao controle de versão digite o comando > git add .

1. Passo

Vamos verificar comando > git status

Confirmar as informações criando um commit comando > git commit -m “criado sistema de login”

1. Passo

Supondo que o código este pronto, agora vamos uni-lo ao branch master digitar o comando > git checkout master

1. Passo

Primeiro vamos atualizar o código da branch master > git pull

1. Passo

agora vamos atualizar a versão master usando o comando > git merge sistema-de-login

1. Passo

Agora vamos enviar as alterações para o controle de versão comando > git push

1. Passo

Agora vamos no GitHub para verificar as alterações

E FIM

PULL REQUEST UM MERGE DEMOCRÁTICO

Quando você termina de fazer a sua alteração na sua branch e você precisa passar para a branch master, mais para isso você precisa que outro programador verifique se o seu código está correto. Com isso utilizamos o comando pull request

Vamos utilizar a branch sistema-de-login para exemplificar

1. Entrar na branch sistema-de-login comando > git checkout sistema-de-login
2. Vamos alterar o código incluindo – “digite sua idade”
3. Incluo a alteração no arquivo
4. Agora vamos adicionar comando > git add .
5. Agora vamos fazer o commit comando > git commit -m “adicionado idade ao sistema de login”
6. Agora vamos enviar para o controle de versão comando > git push
7. Agora vamos entrar no GitHub e abrir o Pull request e margiar a branch sistema-de-login com a branch master
8. Verificamos que alguém da equipe vai identificar a alteração feita da idade e vai verificar se o código esta correto. Caso o código esteja correto ele vai confirmar margiando para a branch master e confirmando e FIM

GITIGNORE

Quando você tem um arquivo ou pasta que você não quer enviar para o controle de versão, você utiliza o gitignore.

Exemplo:

Vamos supor que temos uma pasta que se chama Segredos e dentro dela tenha um arquivo texto com as senhas e não deve subir para o controle de versão.

Vamos digitar o comando > touch .gitignore – com isso vai se criar uma pasta dentro do arquivo onde você vai descrever o nome da pasta ou arquivos que você não queira que suba para a versão de controle.

Com isso a pasta gitignore.txt vai controlar os arquivos ou pasta para não serem enviadas para o controle de versão.

Porém o arquivo gitignore.txt deve ser adicionado comando > git add . e depois fazer o commit comando > git commit “adicionando arquivo gitignore” e enviar para o controle de versão por ser um arquivo comando > git push

OBS.: Normalmente o arquivo gitignore fica dentro da branch master

CONSIDERACOES FINAIS

|  |  |
| --- | --- |
| git init | Inicializa um novo repositório |
| git add . | Adiciona os arquivos atuais ao próximo commit |
| git status | Verifica o status atual dos repositórios git |
| git commit -m “msn do commit” | Cria um novo commit com um msn |
| git push | Envia as atualizações para a nuvem na branch atualmente ativa |
| git branch | Permite listar e ver qual branch está ativada atualmente |
| git checkout nome-da-branch | Permite mudar para a nova branch |
| git checkout -b “nome da branch de origem” “nome da nova branch” | Permite mudar e criar uma nova branch com base na outra |
| git merge “branch a receber merge” | Permite fazer a merge da branch ativa atualmente com outra branch |
| git pull | Atualiza a branch atualmente ativa |