DIO – TQI Fullstack Developer Bootcamp

CURSO: INTRODUÇÃO AO Git E GitHub

Otávio Reis Perkles – Frontend Developer Foton

ENTENDENDO O QUE É O Git E SUA IMPORTÂNCIA

Ex. Um repórter que precisa entregar uma matéria. Faz um rascunho e submete a algum revisor, que retorna um arquivo “rascunho 1” com sugestões de alterações. Depois faz as alterações e renomeia para “rascunho 2” e envia para um editor, que retorna um outro arquivo “rascunho 3” com novas sugestões para alterações. O repórter faz as alterações e salva como “artigo” para envio para aprovação final, que pode ser ou não aceito e voltar para novas alterações, e assim por diante, até chegar ao artigo final para publicação.

O Git foi criado em 2005 como sistema de versionamento de códigos distribuídos, por Linus Torvalds, criador do Linux, de forma colaborativa, com pessoas de diversos lugares do mundo, e o Git segue o mesmo sistema, de poder compartilhar os códigos com outras pessoas.

Ele não é o primeiro sistema desse tipo, mas nasce da insatisfação do Linus com o uso de outros sistemas; então ele resolve desenvolver um sistema próprio, que melhor atendesse as necessidades dos programadores, assim, surgindo o Git.

Basicamente será um local onde se pode trabalhar e armazenar os seus códigos na máquina local, para depois compartilhar com outras pessoas.

Importante saber que Git (é aberto) é diferente de GitHub (que é da Microsoft). São programas que se conversam, mas que não são a mesma coisa. Enquanto o primeiro está instalado na máquina local, o segundo está remoto.

Os benefícios em aprender essa matéria: controle de versão; armazenamento em nuvem dos códigos; trabalho em equipe; melhorar seu código (expõe para todos interagir); e reconhecimento (interação social, que pode dar destaque ao seu trabalho).

NAVEGAÇÃO VIA COMMAND LINE INTERFACE E INSTALAÇÃO DO Git

Comandos básicos para um bom desempenho no terminal

Git não tem uma interface gráfica. A tecnologia é por linha de comando.

Máquinas com Windows terá diferenças para utilizar, já que o Git provém da Linux.

No botão Windows, digitar CMD, vai abrir o C:\Users\>

Listar todas as pastas: comando “dir” e teclar enter.

Comando: “cd /” enter, sobe para o diretório C.

Agora se usar o dir, vai listar as pastas no diretório C.

Entrar nas pastas: Uma das pastas dentro de C é a pasta Windows, para entrar nesta pasta basta dar o comando “cd Windows” e vai entrar na pasta Windows.

Depois comando dir, vai listar tudo que estiver na pasta Windows.

Para voltar um nível: comando “cd ..” e volta para o diretório C.

Para limpar o terminal: comando “cls” para limpar a tela.

Tab é função para auto completar. Basta digitar o início do nome do arquivo que quer buscar dentro da pasta e teclar o tab que já completa o nome do arquivo.

Criar diretório dentro da pasta: “mkdir workspace” (vai criar dentro do diretório onde estiver; se não retornar erro é porque deu certo; o comando é mkdir e seguido do nome do novo diretório).

Se der dir vai listar essa pasta.

Cls para limpar tela.

cd wo+tab – busca a pasta workspace.

Criar arquivo dentro da pasta: >echo hello > hello.txt (>echo serve para buscar se já existe um arquivo com mesmo nome) e o restante serve para criar o arquivo novo.

Para sair: cd ..

Para listar: dir

Para deletar dentro da pasta: del workspace (apenas para arquivos que estão dentro da pasta e não para o repositório).

Vai perguntar se quer deletar: S (para confirmar).

Para deletar o diretório: rmdir workspace /S /Q

Realizando a instalação do Git

Link do site oficial do Git para instalar no Windows: git-scm.com

Entendendo como o Git funciona por debaixo dos panos

Tópicos fundamentais para entender o funcionamento do Git

SHA1

Secure Hash Algorithm (Algoritmo de Hash Seguro), é um conjunto de funções hash criptográficas projetadas pela NSA (Agência de Segurança Nacional dos EUA).

A encriptação gera um conjunto de caracteres identificadores de 40 dígitos.

Toda vez que fizer alterações, será gerado um novo arquivo diferente.

Sempre aparece no projeto uma forma curta para representar o arquivo, normalmente com os 7 primeiros caracteres.

Um mesmo arquivo idêntico sempre gera o mesmo SHA1, se for alterado qualquer mínimo dado, gera um SHA1 novo.

Objetos Fundamentais – Internos do Git

BLOBS

É uma “bolha”, um objeto com conteúdo dentro dele, metadados dentro.

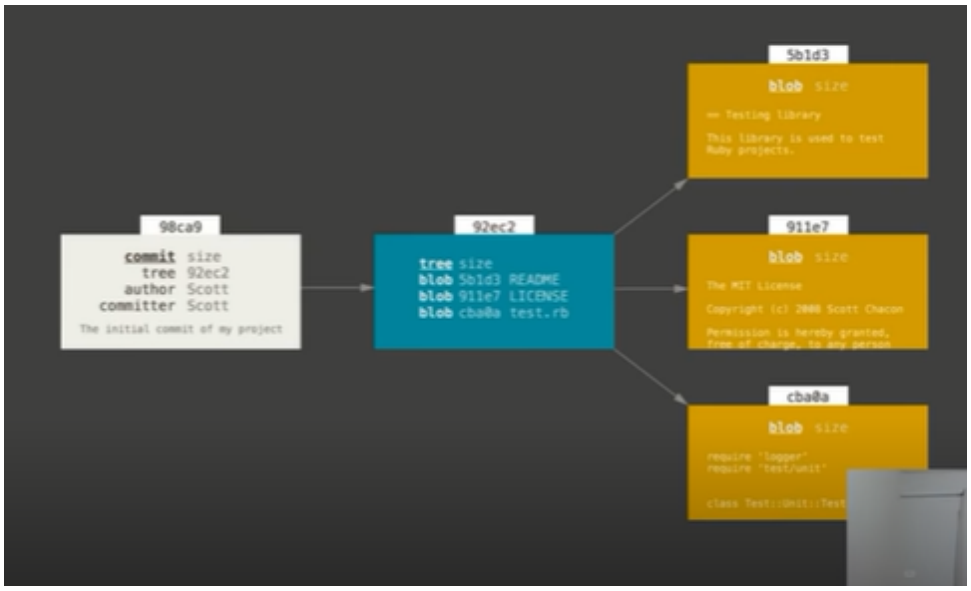
TREES

As “árvores” armazena BLOBS. Estruturam e apontam para outros objetos, inclusive outras árvores.

COMMITS

Objeto que junta toda a operação realizada. Aponta para Tree, para Parente, para Autor, para Mensagem.

Esse objeto também tem um “carimbo” de tempo, do momento quando for criado. Também tem seu SHA1 (se alterar qualquer arquivo, ele também será alterado). Assim criando uma cadeia em sequência, por onde pode ser rastreado o passo a passo das alterações realizadas.



Sistema distribuído e Seguro

Como o Commit não pode ser alterado, sem que seja gerada um arquivo novo diferente, assim, em qualquer computador onde estiver esse arquivo ele será igual.

Por esse motivo ele é um sistema Seguro, pois, mesmo que aconteça alguma coisa com o local de armazenamento, por exemplo, a nuvem onde estiver armazenado, estará também em 40, 50 ou centenas de computadores e sempre iguais, assim pode ser obtido e recuperado facilmente.

Chave SSH e Token

Para maior segurança cada usuário tem sua identificação e uma chave de acesso e token para validar seus dados que são armazenados na plataforma.

A chave SSH é conexão segura e encriptada entre duas máquinas. Uma chave pública e uma privada. A máquina já fica configurada para isso, depois de realizado o procedimento.

Para isso, no logar site do GitHub, clicar na sua imagem (canto superior direito), ir em Setting, na lista à esquerda, selecionar SSH and GPG keys. No botão verde New SSH key, colocar o título e a chave.

Para gerar o par de chaves abrir o Git Bash e dar os comandos: ssh-keygen –t ed25519 –C conti.rodrigues@hotmail.com

Assim será gerada as chaves no Caminho: C/Users/User/.ssh/id_ed25519.pub

Pede para colocar uma senha. E pronto.

Comando: cd /c/Users/User/.ssh/ e ls (aparece as duas chaves).

Comando: cat id_ed25519.pub (aparece a chave pública).

Basta copiar essa chave pública e colar na plataforma do GitHub e clicar no botão verde Add SSH key.

Pede a senha e pronto, já aparecerá na lista de chaves SSH.

No Git Bash, dentro da pasta /.ssh dar o seguinte comando: eval $(ssh-agent –s) e enter. Vai gerar um Agent pid, um projeto rodando em plano de fundo.

Comando: ssh-add id_ed25519 (a chave privada) e pede a senha e já adiciona a chave para ficar automático o reconhecimento quando clonar os repositórios.

PRIMEIROS COMANDOS COM Git

Iniciando o Git e criando um commit

Iniciar o Git Bash

- git init

- git add

- git commit

Todo comando leva o git no início.

Para nossas atividades vamos criar o diretório “workspace” dentro de C no Windows.

Para iniciar o Git, posso ir diretamente na tela do Windows onde está salva a pasta e clicar com botão direito do mouse, na opção “Git Bash here”, que já abre o Git direto com essa pasta do C, não precisando assim navegar até esse diretório que desejo trabalhar.

Para listar o conteúdo no Git usa o comando “ls” (list – igual no Linux).

Para entrar na pasta, o comando é o “cd nome/” – no caso, cd workspace/.

Para limpar a tela, basta dar um CTRL+L.

Comando: pwd (para dar o caminho completo da pasta onde está).

Para criar um diretório novo o comando é igual: “mkdir nome” (no caso, mkdir livro-receitas).

Para entrar na pasta: cd livro-receitas/

Para iniciar o repositório o comando: git init (cria um “master”).

Para listar pastas ocultas: ls –a (aparece todas as pastas).

Agora entrar na pasta: cd .git/ e listar ls

Para voltar um nível é igual: cd .. (no caso, volta para pasta livro-receitas).

Se for a primeira vez que utiliza o Git ele exigirá uma configuração, que será um name e um e-mail.

Para isso o comando: git config –global user.email “conti.etc” (usar as aspas)

Comando para usuário: git config –global user.name ContiRodrigues

Importante cadastrar o mesmo que se utiliza no GitHub, para que não tenha divergência de nomes do autor.

Adicionar um arquivo

Basta entrar na pasta “livro-receitas” no Windows e criar um arquivo, clicando com botão direito do mouse, novo, escolher qualquer editor (usar o Typora – baixar no site), nomear o arquivo e apagar a extensão e colocar a extensão “.md” (Markdown). No caso, vamos criar o arquivo “strogonof.md” dentro da pasta livro-receitas.

Basta dar dois cliques no arquivo e abrir para digitar o texto, no caso, a receita.

Baixar o editor de texto Typora, que é compatível com o Markdown. Abrir com Typora.

Link: https://typora.io/

O Markdown tem a mesma estrutura das Tags do HTML para níveis de textos:

# Título nível 1	(Título nível 1)
## Título nível 2	(Título nível 2)
### Título nível 3	(Título nível 3)
#### Título nível 4	(Título nível 4)
##### Título nível 5	(Título nível 5)
###### Título nível 6	(Título nível 6)

Digitar # Strogonof de Frango – automaticamente o Typora já edita com o tamanho da letra nível 1.

Para usar o negrito, basta usar **palavra** e já edita com negrito.

Para itálico, basta usar _palavra_ e já edita com itálico.

Pode usar emotions, digitando dois pontos e o nome. No caso, :chiken e já aparece a galinha.

Colocar os ingredientes e modo de preparo. Para esses dois títulos usar ###.

Nos ingredientes usar lista não ordenada: (-) menos espaço e enter e já irá criar.

No modo de preparo usar lista ordenada: 1. 2. 3. Etc.

Salvar o arquivo no Typora e fechar.

Voltar no Git e dar o comando: git add *

Em seguida: git commit –m “commit inicial”

Ciclo de vida de arquivos no Git – Passo a passo

Git init – comando para iniciar/criar o repositório na pasta.

Tracked – arquivo rastreado pelo Git.

Untracked – arquivo que o Git não tem conhecimento dele.

Nos arquivos Tracked podem ser:

Unmodified – arquivo ainda não modificado.

Modified – arquivo que já foi modificado.

Staged – conceito chave (ex. parte de trás do palco – o back stage). Arquivo se preparando para fazer parte de outro agrupamento.

Quando uso o comando git add – o arquivo strogonof estava Untracked. E foi para Staged.

O Git compara o SHA1, e muda dos arquivos Unmodified para Modified.

Quando coloca o arquivo no Staged, e dá comando commit passa a ser Unmodified.

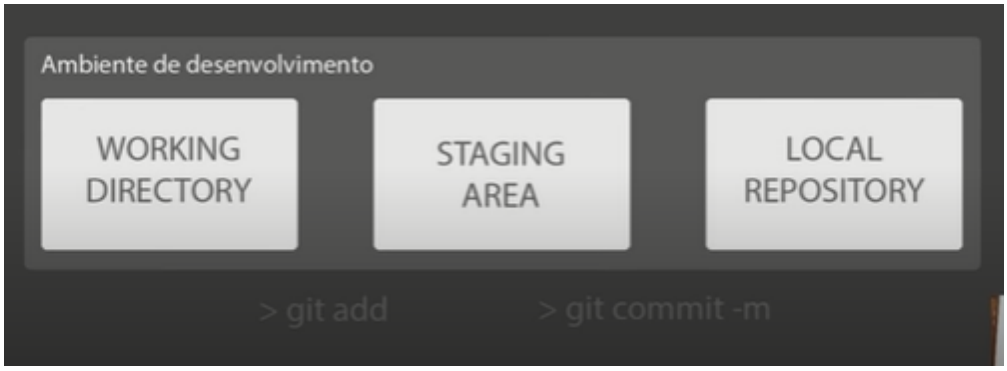
É um ciclo entre esses três estágios. Volta para Unmodified quando comita, se abre ele e faz modificações vai para Modified, e se adiciona a um commit, ele vai primeiro para um Staged e quando escreve o commit ele volta para Unmodified.

Dois ambientes: o de Desenvolvimento Local (na máquina) e o Servidor (GitHub).

As mudanças feitas na máquina não refletem imediatamente no Remote Repository (GitHub).

No ambiente de desenvolvimento há o diretório de trabalho (Working Directory), e há a área de staging (Staging Area). Quando faz um commit passa a integrar o Repositório Local (Local Repository), que poderá ser empurrado para o Repositório Remoto.

Assim, todo o repositório local é composto por commits, que podem ser empurrados para o Remoto.



Criando nova pasta e movendo arquivos

Comando git status – traz o status dos arquivos.

Para criar uma nova pasta dentro da livro-receitas: mkdir receitas

Para mover um arquivo, comando: mv strogonof.md ./receitas/ (moveu o arquivo strogonof para dentro da pasta receitas).

Agora dando o comando git status, aparece o arquivo strogonoff como deletado, pois foi enviado para uma pasta que não estava criada quando demos o commit anteriormente, foi para a pasta receitas que criamos agora, e que está como Untraked.

Assim, vamos adicionar o arquivo novamente e a nova pasta ao commit.

Para isso, o comando: git add strogonof.md receitas/ (altera o status para Staged).

Perceba que adicionamos o arquivo e a pasta específicos, se fossem diversos poderia ser dado o comando git add . ou git add * (ambos servem para todos os arquivos alterados ou novos).

Para verificar o status: git status (e vemos que o arquivo e a pasta estão para ser comitados).

Comitando: git commit –m “cria pasta receitas, move arquivo para receitas”.

Se der comando: git status não aparecerá nada mais para comitar.

Agora se entrarmos nos arquivos ou pastas e modificarmos, eles entrarão automaticamente para a condição de Unmodified, ou seja, retorna ao ciclo até chegar para ser comitado novamente.

Criando o README

Esse arquivo tem a extensão .md (Markedown). Ele é muito importante e é considerado uma praxe, uma boa prática, pois serve como uma apresentação do projeto, uma sintaxe do que será trabalhado.

No caso, vamos criar um index para sabermos quais receitas temos na pasta.

Comando: echo > README.md

Se der git status vai retornar que há arquivo do tipo Untracked, ou seja, desconhecido, não rastreado.

Abrir o arquivo no Typora. Fazer a capa do livro, apresentação, e listar as receitas.

Voltamos no Git e comando git status.

Comando git add * (que pega todos os arquivos que estão no Untracked para Staged).

Para verificar: ls para listar, e um git status.

Comitar: git commit –m “adiciona index”

E será adicionado nosso arquivo de README.md ao projeto.

INTRODUÇÃO AO GitHub

Configurações da plataforma

Criar nova conta no site do GitHub.

Link: https://github.com

Usuário: ContiRodrigues

Senha: XXXXXXX

Importante que o nome e e-mail no GitHub seja o mesmo cadastro no Git, para evitar divergência de autor.

Caso necessite alterar no Git (para deixar igual e facilitar o trabalho):

Comando: git config –global user.email “conti.etc”

Comando: git config –global user.name “ContiRodrigues”

Cadastro da Chave SSH

Para cadastrar a Chave SSH do Git no GitHub:

Chamar o arquivo: cd /c/Users/User/.ssh/

Listar os arquivos: ls aí vai aparecer id_ed25519.pub

Comando: cat id_ed25519.pub (pois a chave que irá expor sempre será a chave pública, e a privada fica na máquina para fazer a confirmação quando solicitada).

Assim, vai aparecer a chave pública, basta copiar e colar no GitHub.

No GitHub clicar na Foto (canto superior direito), Setting, depois em SSH and GPG keys, e New SSH (botão verde). Colar a chave no campo Key e dar um nome no campo Title.

Vai pedir a senha e já cria a chave SSH.

No Git dar o comando: eval $(ssh-agent –s)

Dar o comando: ssh-add id_ed25519 (ou seja, já deixo configurado a minha chave privada para toda vez que chegar uma solicitação o sistema já realiza a descriptografia).

Para confirmar basta digitar a senha cadastrada.

Clonando o repositório do GitHub para o Git com chave SSH

Para clonar do GitHub, basta copiar o SSH e no Git dar o comando: git clone e o caminho SSH gerado.

Assim, o repositório será criado no automaticamente no Git.

Criar Token GitHub para assinatura pessoal

No site do GitHub, clicar em Developer setting, depois em Personal acess tokens, e em Generate new token.

Escolher o Nome e o prazo: 30, 60 ou 90 dias.

Selecionar a opção “Repo”.

Salvar em local confiável essa chave, para copiar sempre que precisar.

Clonando o repositório do GitHub para o Git com token

Esta chave token serve para clonar do GitHub para o Git com o HTTPS.

Assim, basta copiar o https no GitHub e no Git dar o comando: git clone e o https.

Vai pedir o token. Basta copiar e colar. E pronto, já será clonado no Git o repositório do GitHub.

Criando um novo repositório no GitHub

No GitHub, clicar na foto no canto superior direito, depois em “seus repositórios”, e “new”.

Colocar o nome (no caso, “livro-receitas”). E uma descrição (Meu livro de receitas).

Escolhe se quer deixar o projeto público ou privado.

Clicar no botão embaixo: create.

Agora deve copiar o https criado e abrir o Git para colar.

```
https://github.com/ContiRodrigues/livro-receitas.git
```

Esse procedimento serve para empurrar o que está na máquina para o GitHub.

Comando: git remote add origin e o https acima (copiado do GitHub).

Comando: git remote –v (vai listar os repositórios remotos que tiver cadastrados).

No caso, o “origin” é uma convenção, um apelido, para não ter que digitar o https toda vez que precisar chamar.

Se der um git status, veremos que não tem nada para comitar.

O comando: git push (literalmente para “empurrar”) mais “origin master”.

Vai pedir a confirmação (no caso, o Token cadastrado, basta copiar e colar – lembrar de verificar se está dentro da validade ou vencido).

Basta recarregar a página no GitHub e já estará disponível o projeto lá.

Importante observar que já traz o Readme.md de cara, que é a apresentação do que está sendo disponibilizado.

Resolvendo conflitos no GitHub

A versão que temos na máquina local, no repositório, corresponde ao que está no GitHub.

Acontece que outra pessoa pode clonar o repositório, e se fizer qualquer mínima alteração, já será um código novo e diferente.

Se essa pessoa empurra o código alterado dela para o GitHub, esta será a versão mais atual do código e será a que prevalece.

Caso tenha continuado trabalhando no código e queira empurrar para o GitHub, dará um conflito, pois o seu pode estar com data de commit mais antigo, então prevalece o já empurrado com data mais nova.

Assim, terá que resolver manualmente os conflitos apontados e empurrar novamente, para que seja a última versão do código e possa prevalecer.

No caso, abri o README e acrescentar nova linha com a receita “Pavê” na sequência de Strogonof.

Comando git status – irá aparecer que o README foi modified.

Comando git add * (para salvar todas as alterações que tiver feito).

CTRL+l para limpar. Listar com ls e git status para verificar.

Agora, depois de adicionado, já temos arquivo pronto para ser comitado (changes to be committed).

Comitar: git commit –m “Adiciona receita pave”.

Agora vamos empurrar para o GitHub: git push origin master (pede o token).

(No nosso caso deu certo, e acrescentou o Pavê no README).

Mas, se outra pessoa tivesse feito alterações, daria um erro.

Para solucionar é necessário dar comando: git pull origin master (“puxar” o que está alterado no GitHub e o Git vai tentar juntar os dois para corrigir as diferenças).

Como baixar um projeto do GitHub para repositório local

Ex. Buscar no GitHub a página do Python, e um repositório chamado “cpython”.

Clicar no botão verde Code. Escolher e copiar a https.

No Git Bash ir para o diretório base (workspace).

Comando: git clone e a https. (https://github.com/python/cpython.git)

Depois de baixar tudo (100%), dar ls e estará lá a pasta cpython/

Comando cd cpython/ para entrar na pasta. Dar um ls. E um ls –a para mostrar repositórios ocultos.

Se dermos um comando: git remote –v (vai listar o repositório origin).