**GIT** – Sistema de controle de versionamento de códigos

**GITHUB** – repositório online/ servidor para guardar códigos/versões

**GIT**

- O git é um software que funciona via command line interface = não tem uma interface gráfica e sim interação por linha de comando

Ubuntu – um subsistema no Windows p usar linux

**INTERAGINDO COM O TERMINAL:**

ABRINDO O PROMPT DE COMANDO DO WINDOWS: pesquisar “cmd” na pesquisa do windows

No Windows: Se digitar “dir” + enter: ele dá uma lista de diretórios na pasta a qual vc está situado

No Linux (ubuntu no Windows): Se digitar “ls” + enter: ele dá uma lista de diretórios na pasta a qual vc está situado

Para entrar dentro de uma pasta : No prompt de comando do windows Digita “cd” + espaço + “o nome da pasta que vc quer explorar” (Ex: cd Windows)

Para retroceder um nível na navegação: cd + ..

Para limpar a tela no prompt: cls

A TECLA TAB tem a função de autocompletar a palavra.

Para criar uma pasta: mkdir + o nome da pasta q vc quer criar. Se der certo, após o enter não se recebe nenhuma msg. Mas se der erro ele vai mostrar.

Para criar um arquivo dentro da pasta: echo > nome.txt

Para deletar o conteúdo que existe dentro de uma pasta de algum diretório: del + nome da pasta. Obs: não vai deletar a pasta e sim o conteúdo que tem dentro dela!

Para deletar uma pasta ou diretório: rmdir + nome do diretório + / S /Q

**ENTENDENDO O GIT**

.A sigla SHA (Algoritmo de Hash seguro) é um conjunto de funções criptográficas projetadas pela NSA (Agencia de segurança nacional dos EUA) – o algoritmo pega o seu arquivo e vai “embaralhar” ele de uma forma muito específica. Essa encriptação gera um conjunto de caracteres identificador de 40 dígitos – uma chave.

**OBJETOS INTERNOS DO GIT**

. Strings em Java são objetos que contém uma cadeia de caracteres. São imutáveis, portanto, uma vez criadas não podem ser alteradas. Caso ocorra uma concatenação de Strings, um novo objeto é criado, e o antigo, automaticamente removido. Em programação, cada símbolo armazenado na memória é representado por um valor numérico. Uma variável declarada com tipo de dado cadeia geralmente armazena um número pré-determinado de caracteres. **String = cadeia**

. São responsáveis pelo controle de versionamento no Git

BLOBS: Bloco básico de composição. Guarda o sha1 do arquivo, mas não guarda o nome do arquivo. É um objeto que encapsula diretórios, apontando para arquivos que contenham diretórios. Contém metadados do git: tipo, tamanho do arquivo...

TREES: Armazenam blobs e contém também metadados. Guarda o nome do arquivo. Responsável por montar toda a estrutura de onde estão os arquivos. Apontam para blobs ou para outras trees (isso porque em um sistema operacional é normal que um diretório contenha outros diretórios).

As trees também tem um sha1 desses metadados.

O objeto trees está diretamente relacionado às blobs. Ou seja, se muda um arquivo na blobs, automaticamente mudará toda a estrutura e encriptação na trees.

COMMITS: O commit é um objeto que dá sentido a tudo, pois junta tudo e dá sentido para a alteração que você está fazendo. O commit aponta para uma árvore, aponta para um parente (o ultimo commit realizado antes dele, aponta para um autor e para uma mensagem também).

O commit também tem um timestamp (Carimbo de tempo) – leva a data/hora de quando ele foi criado

Os commits também possuem um sha1, uma encriptação de toda essa informação. Então se você alterar uma blob, altera também os metadados da tree, que por sua vez altera a estrutura inteira, incluindo os commits!

O git é um sistema distribuído seguro!

**CHAVES SSH E TOKENS**

CHAVE SSH – uma forma de estabelecer uma conexão segura e encriptada entre 2 máquinas

1. Executar os comandos no GIT BASH para gerar a chave SSH no seu perfil do GITHUB:

ssh-keygen –t ed25519 –c seuemail@gmail.com

1. Ele gera 2 chaves: uma pública e uma privada (que termina com .pub)
2. Digitar cd /c/users/seudiretorio/.ssh/ no prompt de comando do GIT BASH para ir até a pasta que contém a chave criada
3. Digitar comando “ls” para listar o que tem dentro da pasta
4. Digitar comando “cat id\_ed25519.pub” para aparecer a chave pública
5. Quando aparecer a chave pública, copiar e colar na parte CHAVE SSH do GITHUB
6. Inicializar o SSH AGENT (que vai rodar em plano de fundo. É uma entidade responsável de pegar as chaves e lidar com elas) com o seguinte comando: “eval $(ssh-agent –s)”. Daí vai surgir um número, é como se ele estivesse startando um projeto
7. Passar a chave privada para o ssh agent usando o comando: “ssh-add id\_ed25519”

COMANDOS GITBASH:

* pwd = mostra o caminho completo p chegar na pasta que vc está atualmente
* ls = listar o que contém na pasta que vc se encontra
* cd + nome da pasta = entra na pasta que vc quer
* ctrl + l = limpar a tela
* git init = iniciar o git
* git add = iniciar o versionamento
* git add \* = inicia o versionamento de tudo que existe dentro da pasta
* git commit = criar um commit
* cd + .. = retroceder um nível na navegação
* git status = saber o status dos arquivos: unmodified, modified, staged..
* mv “nome do arquivo” ./”nome da pasta”/ = mover um arquivo para uma pasta
* git remote –v = lista a lista de repositórios remotos que eu tenho cadastrado
* git clone = pegar o repositório e clonar para a sua maquina
* git pull origin master = puxa o repositório mais atual que está no github para a sua maquina
* git push origin main = empurra p repositório remoto as alterações feitas na maquina
* git rm –rf nomedapasta = remove

FLAGS GITBASH:

* -a = mostra pastas/arquivos ocultos dentro do diretório que vc se encontra
* Git commit – m “msg” = pega os arquivos que estavam staged, envelopa uma mensagem, dando significância a eles
* Git add + nome do arquivo ou \* ou . = adiciona os arquivos para a staging área. Estão no ponto para serem commitados depois.

**PRIMEIROS COMANDOS COM O GIT**

- INICIAR O GIT ( comando: git init)

- INICIAR O VERSIONAMENTO (comando: git add)

-CRIAR UM COMMIT (comando: git commit)

- GIT IGNORE serve para não ir arquivos desnecessários quando for subir o projeto para o github

* Quando estamos lidando com um terminal, se coloca sempre o nome do programa na frente antes do comando, ex: “git init”

1. Criar um arquivo “LIVRO DE RECEITAS” dentro da pasta WORKSPACE pelo GITBASH:

- comando “cd + workspace” p/ ir p pasta workspace

- comando “mkdir + livro-receitas” p/ criar uma pasta dentro da pasta workspace chamado livro-receitas

- comando “ls” p/ listar oq existe na pasta e confirmar que foi criada a nova pasta livro-receitas

1. Iniciar o GIT dentro da pasta livro-receitas:

- control + l p/ limpar a tela do terminal

- cd + livro-receitas p/ ir até a pasta livro-receitas

-“git init” p/ começar a iniciar o git e versionar o código

- “ls” p/ visualizar o que tem dentro da pasta

- não vai aparecer nada, apesar de ter um arquivo gerencial chamado .git na pasta. Mas não aparece pq arquivos que começam com ponto são arquivos ocultos.

-P/ visualizar os arquivos ocultos dentro de um diretório usa-se a FLAG –a:

- “ls –a” p/ listar arquivos ocultos na pasta e assim irá aparecer o .git

1. Criar um arquivo dentro da pasta livro-receitas:

-p/criar um arquivo, antes deve-se fazer uma configuração inicial (se for a primeira vez usando o git)

- “git config –global user.email [seuemail@gmail.com](mailto:seuemail@gmail.com) p/ configurar o email

- “git config –global user.name seuusername p/ configurar o username

- Criar um arquivo Markdown (uma forma mais humana de escrever em html) na pasta: Criei um arquivo .txt e depois converti pra .md – strogonoff.md

1. Commitar o arquivo strogonoff.md

-git add \*

-git commit –m “commit inicial”

1. Criar uma pasta “receitas” dentro da pasta “livro-receitas” e mover o arquivo “strogonoff.md” para a pasta “receitas”:

- mkdir receitas p/ criar uma pasta chamada receitas dentro da pasta “livro-receitas”

- mv strogonoff.md ./receitas/ p/ mover o arquivo “strogonoff.md” para a pasta “receitas”

1. Verificar o status dos arquivos na pasta “livro-receitas”, mover o arquivo criado recentemente “receitas” para o status de staged e depois commitar :

-git status p/ verificar o que ainda precisa ser commitado

- git add strogonoff.md receitas/ p/ adicionar esses arquivos novos no staged

- git status p/ verificar como está o status dos arquivos

-git commit –m “cria pasta receitas, move arquivo para receitas”

**O CICLO DE VIDA DOS ARQUIVOS NO GIT**

* GIT INIT: Ao usarmos esse comando ele inicializa um conceito chamado repositório
* GIT ADD: Usamos um arquivo que estava untracked (pois tínhamos acabado de criar ele – strogonoff.md) que se moveu direto para staged. Os arquivos que estão “staged” estão na verdade se preparado para serem um arquivo commit, estão esperando para entrar em ação.
* COMMITED: Qdo o arquivo sai do “staged” e é commitado, ele volta pro Unmodified (aguardando novas modificações).. Portanto é um ciclo. Unmodified – Modified – Staged – Commited – Unmodified... Os arquivos ficam transitando entre o working directory e a staging área.
* Quando vc faz um COMMIT ele passa a integrar o seu repositório local. E o seu repositório local por sua vez pode ser empurrado para o repositório remoto (github)
* O COMMIT é uma espécie de “foto” do nosso código naquele momento
* MERGE CONFLICT: Qdo alguém modifica/edita o arquivo no repositório remoto e o que tem no repositório da sua maquina não é igual ao que está lá. Então tem que fazer o comando “ git pull origin master” p/ atualizar. – o github tenta integrar os 2 repositórios fazendo com que fiquem na mesma versão. Abre-se as versões, vê qual está correto ou o que tem que ser modificado e depois commita novamente.