**Git & GitHub**

**Git** é um sistema de versionamento (ou de controle de versão) de códigos distribuído. É um software que serve para funcionar de maneira colaborativa, ou seja, com o Git é possível que um mesmo arquivo seja modificado por duas pessoas e que ambas as alterações sejam salvas e nenhuma delas fiquem sobrescrita. Ajuda criando e monitorando diferentes versões de um código. Para o funcionamento de modo coletivo, o Git utiliza o conceito de ramificação (branch) em que se pode possuir várias ramificações em que arquivos podem ser alterados livremente ser impactar outras ramificações. Com isso, é possível trabalhar com ramificações diferentes para cada desenvolvedor que por fim no momento de juntar esses trabalhos o Git utiliza o conceito de mesclagem (merge) para isso.

Sendo o Git um programa destinado a cuidar do versionamento dos códigos, o **GitHub** é um repositório online onde será armazenado o código, ou seja, funciona como hospedador para todo o armazenamento das diferentes versões de códigos que permite o gerenciamento dessas mais diversas versões de um arquivo. Nessa plataforma cada projeto funciona como um repositório.

**Graphical User Interface – Interface gráfica do usuário (GUI)**

Tipo de interface que permite a interação através de elementos gráficos. Automatizam a interação por linhas de comando e as transforma de interfaces gráficas.

**Command Line Interface – Interface de linhas de comando (CLI)**

Programa que não utiliza interface gráfica, possuindo sua utilização por linhas de comando.

Comandos de terminal

DIR – mostra uma lista de diretórios contidos na pasta que se encontra.

CD (change directory) – possibilita navegar entre as pastas

CD .. – retrocede uma pasta

MKDIR (make directory) – mkdir workspace - cria um diretório

ECHO – printa de volta no terminal uma frase passada

ECHO hello > hello.txt

(‘>’ é um redirecionador de fluxo, em que ele vai pegar o output e colocará em um arquivo.)

DEL – deleta arquivos presentes em um repositório, porém ser deletar juntamente este mesmo repositório.

LS -A – lista arquivos ocultos.

CTRL + L – limpar o terminal

RMDIR (remove directory) – remove o repositório junto com os diretórios presente nele.

|  |  |
| --- | --- |
| Windows | Linux |
| - CD | - CD |
| - DIR | - LS |
| - MKDIR | - MKDIR |
| - DEL / RMDIR | - RM - RF |

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). Algoritmo de encriptação utilizado pelo Git para poder identificar os arquivos de uma forma segura e rápida. A encriptação gera conjunto de caracteres identificador de 40 dígitos. De modo simplificado, é uma forma curta de representar um arquivo – de identificá-lo.

Passar um arquivo para o algoritmo de encriptação SHA1:

openssl sha1 texto.txt

Esta é uma forma eficiente de conseguir identificar alterações realizadas em arquivos.

Objetos Git

3 tipos básicos de objetos do Git responsáveis pelo versionamento de códigos:

**• Blobs**

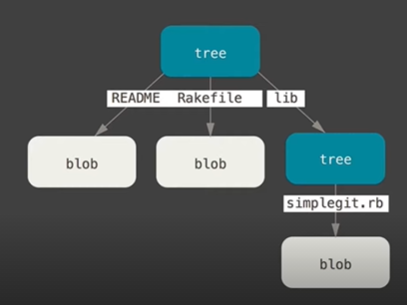
Blob possui metadados em sua formação e mantém arquivos guardados, armazenando: tipo, tamanho e conteúdo – estrutura básica

Blob tamanho \0 ‘conteúdo’

Isso quer dizer que o Git guarda os arquivos gerando a encriptação SHA1, mas também armazenando juntamente metadados nesses objetos.

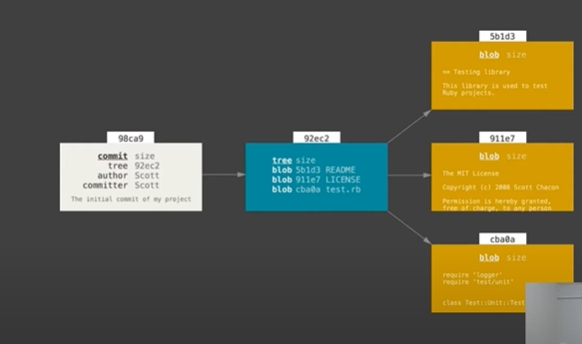
**• Trees**

Por sua vez, tree armazena a aponta os tipos de Blobs diferentes (bloco básico de composição). Metadados desse objeto: blob apontado, SHA1 e nome do arquivo. A árvore é responsável por montar toda a estrutura de onde se localiza os arquivos. Trees pode apontar também para outras trees, já que diretórios dentro de um sistema operacional podem conter outros diretórios. Por sua vez, as árvores também possuem um SHA1 dos metadados delas, com isso, os blobs possuem o SHA1 dos arquivos e as árvores que apontam para esses blobs possuem um SHA1 para seus metadados.



**• Commits**

É o objeto responsável por juntar tudo, no qual o commit aponta para uma tree, para um parente (último commit realizado antes dele), autor, mensagem. O commit funciona como forma de explicação para as alterações e arquivos localizados em blobs e trees. Os commits também possuem seu SHA1, fazendo com que alteração seja em uma blob, tree ou commit, vai fazer com que todos os SHA1 presentes sejam alterados. Devido ao commit apontar também para parentes, é possível montar uma linha do tempo, sendo possível demonstrar alterações ou a veracidade de inalterações presentes no commit. Isso tudo faz com que o Git seja muito confiável.



Markdown (.md) – linguagem simples de marcação. Converte o texto digitado em HTML válido. Forma humanizada de se programar HTML sem necessariamente conhecer a programação HTML.

Comandos Git

GIT INIT – cria um repositório do Git e possibilita que ele comece de fato a gerenciar os arquivos e versionar dentro do diretório que se encontra.

Obs: .git é uma pasta oculta que se localiza todo o código do Git e onde ele versiona.

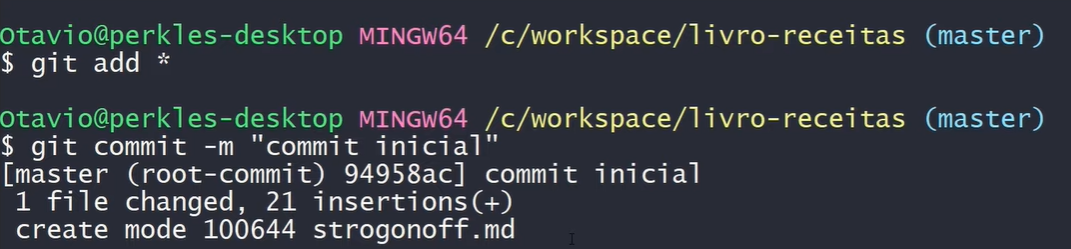
GIT ADD – mover arquivos e dar início ao versionamento

GIT COMMIT – criar commit

git commit -m “mensagem sobre o commit”

GIT STATUS – dá um status dos arquivos, dizendo se o arquivo está untracked, modified, unmodified, tracked etc.

Comando para criar um commit



Tracked ou Untracked

Staged: onde fica os arquivos que estão se preparando para fazer parte de um commit (back stage – bastidores na espera para realizar algo).

O comando **git add \*** move o arquivo para Staged, sendo ele untracked ou modified. O commit retorna todos os arquivos passados pelo staged para unmodified novamente.



Usando GitHub

‘git remote add origin (link do repositório)’ – empurra a versão do repositório local para o remoto dando a origem para onde vão esses arquivos. Origin funciona como um apelido para não ser necessário digitar a url toda vez.

‘git remote -v’ – mostra uma lista dos repositórios remotos cadastrados.

‘git push origin master’ – leva os arquivos do repositório local para o repositório remoto.

‘git config --list’ – lista as configurações do Git

‘git pull origin master’ – puxa os arquivos do repositório remoto para o local.

‘git clone git clone (link do repositório)’ – clona todo os arquivos do repositório GitHub para o local.

Failed to push some refs.

Não é possível realizar um push caso seu documento seja mais antigo que o encontrado no repositório remoto. Ocorre normalmente quando alguém dá push antes, sendo necessário pegar os arquivos necessários e mais atualizados e partir deles então editar o que for necessário para assim dar um novo push.

