BOOTCAMP INTER - FRONT

AULA 2 - Estruturas de Dados

São criadas através de algoritmos.

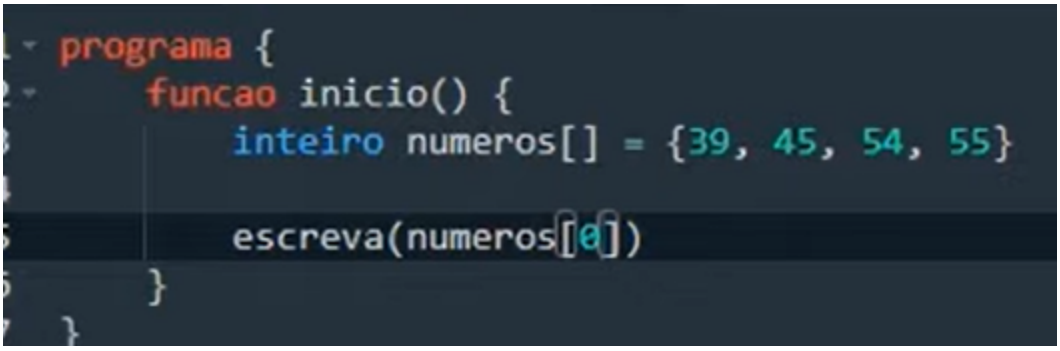
Operações básicas: Inserir dados, excluir dados, localizar um elemento, percorrer todos os itens, classificar (colocar em ordem numérica, alfabética etc)

Principais estruturas: Vetores, Matrizes, Registro, Lista, Pilha,Fila, Árvore, Tabela Hash e Grafos.

VETORES E MATRIZES (arrays)

Variáveis do mesmo tipo e indexada.

nome [2] = 'Fernando', 'Lari' ou nome [ ] = { Lari, fernando, bernado }



Pesquisando no vetor:

```
funcao inicio()
{
    inteiro vetor[] = { 1, 3, 5, 7, 9}  // Cria o vetor com valor
    inteiro numero
    logico achou = falso  // Variável para armazenar o resultado

    escreva ("Qual número deseja procurar? ")
    leia (numero)

    para (inteiro posicao = 0; posicao < 5; posicao++)
    {
        se (vetor[posicao] == numero)
        {
            escreva ("Encontrado na posição: ", posicao, "\n")
            achou = verdadeiro
        }
    }

    se (nao achou)
    {
```

Obs, criação de uma variável boolean.

Para Matriz, 2 for, um para linha e um para coluna.

```
const inteiro TAMANHO = 5

// Cria a matriz
inteiro matriz[TAMANHO][TAMANHO]

para (inteiro linha = 0; linha < TAMANHO; linha++)
{
    para (inteiro coluna = 0; coluna < TAMANHO; coluna++)
    {
        matriz[linha][coluna] = u.sorteia(1, 9) // Atribui

        escreva("[", matriz[linha][coluna], "]") // Exibe o
    }

    escreva ("\n")
```

REGISTRO

Armazenar dados de tipos diferentes.

Campos podem ser acessado por (.)

livro = {preco,nome,autor,ano}

livro.preco

LISTAS = ordem especifica.

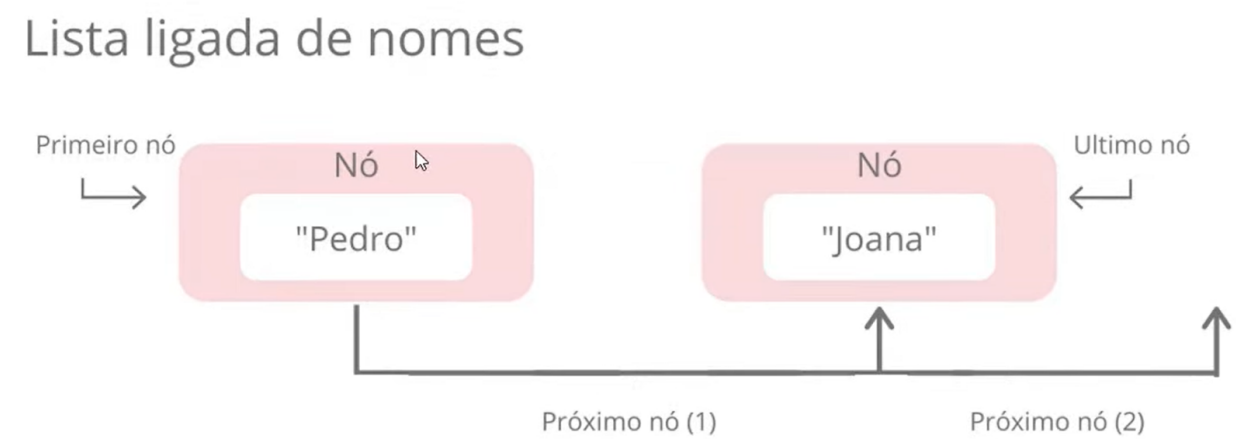
Possui um tamanho ajustado, diferente do array.

Ela nao precisa ser inicializada, pode ser preenchida depois.

Manipulada durante a execução, flexivel, variavel.

⇒ LIGADAS

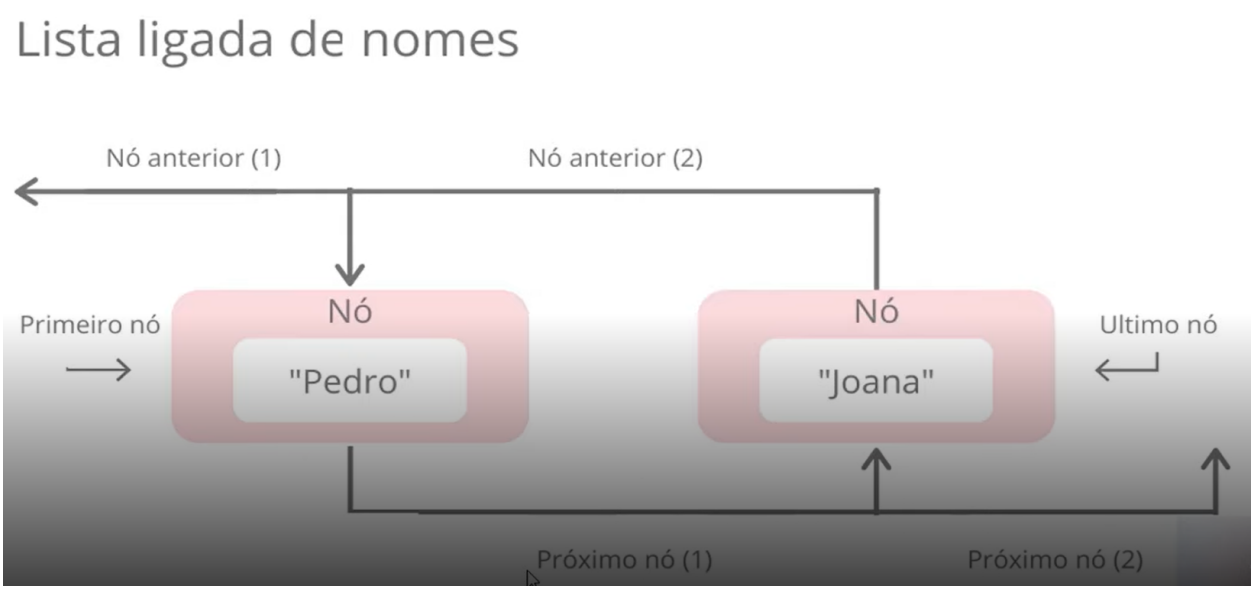
Existe nó, que é como se fosse o índice. Conhece o elemento que esta armazenado e o próximo.,



⇒DUPLAMENTE LIGADAS

Diferença é que duplamente ela é bi-direcional.

Consegue andar para frente e voltar.



PILHAS - STACK

Coleção de elementos e permite acesso a somente um item de dados.

Acesso restrito.

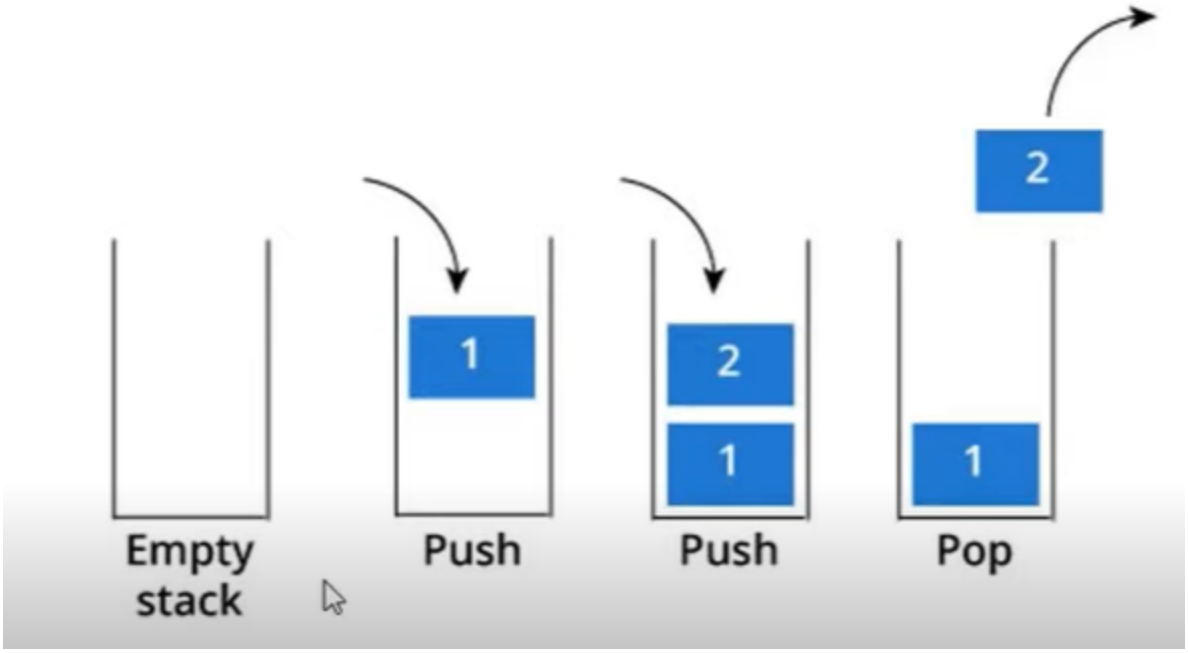
Somente pode ser lido ou manipulado uma vez.

Elementos vao por cima.

⇒ LIFO (Last in First Out)

Ultimo que entra, primeiro que sai. (Sai por cima)

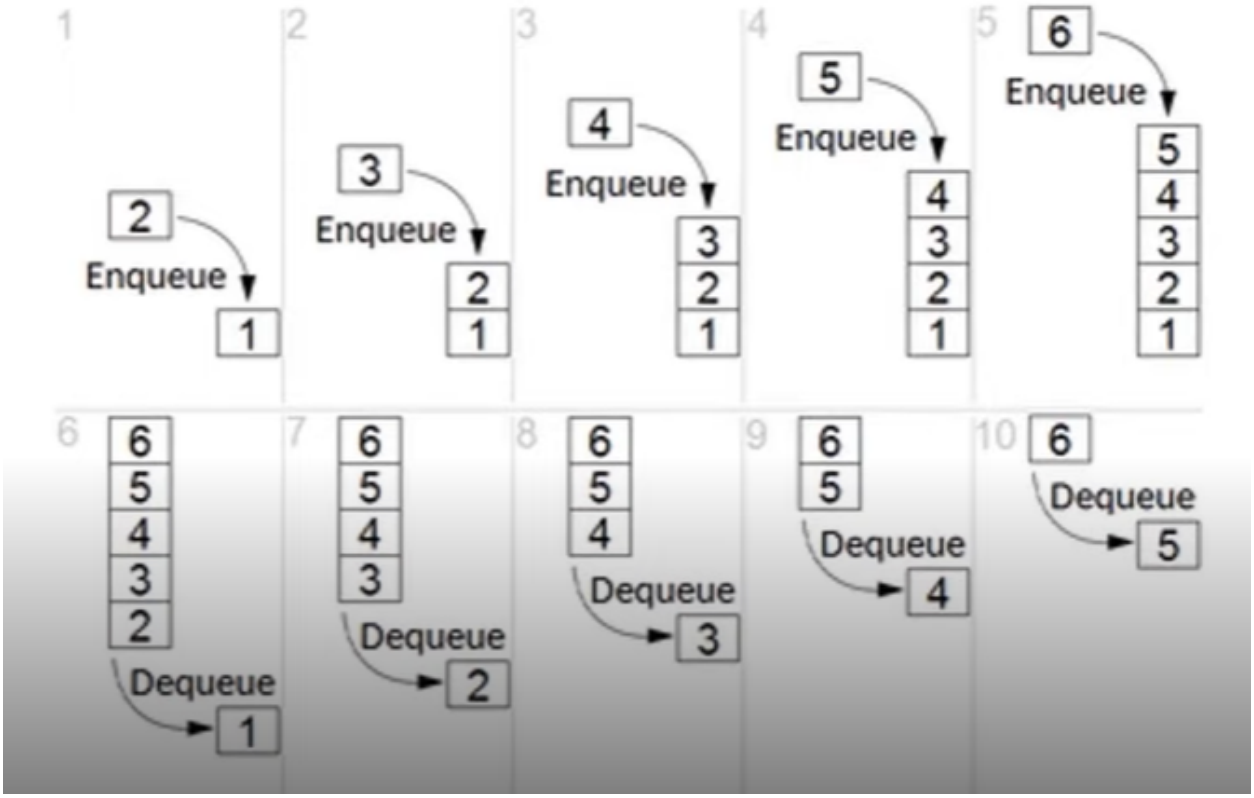
O topo que consegue manusear.



⇒ FIFO (Fist in Fist Out)

Primeiro que entrar, primeiro que sai (sai por baixo)

Manusear o bottom

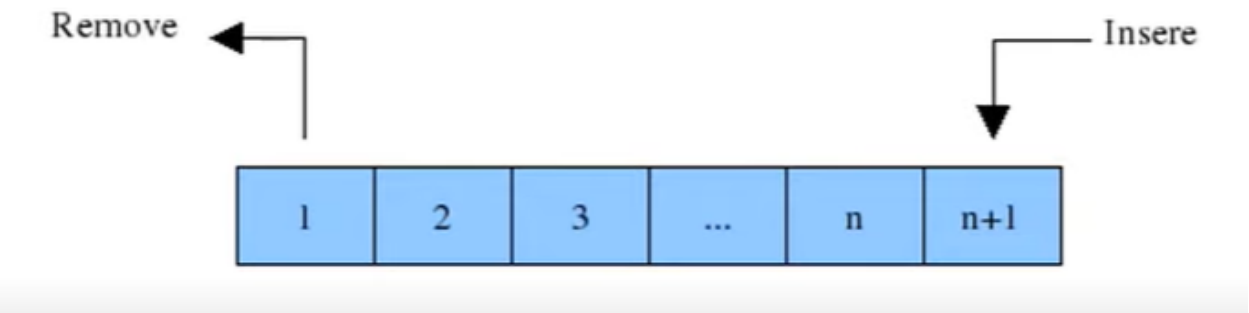


FILAS

Na remoção: sempre o que esta na fila a mais tempo. Parecido com a Pilha FIFO.

Primeiro que entra, é o primeiro a sair.

Fila de banco, lembrar.



Remove esquerda, insere na direita.

ARVORE

Estrutura organizada de forma hierarquica.

Raiz e sub elementos (nós ou folhas).

Facilita a busca.

TABELA HASH (espalhamento)

Usar a função denominada hashing.

Associar valores a chaves, cada valor recebe um código.

GRAFOS (permitem programar relação entre objetos) - mas usado em IA.

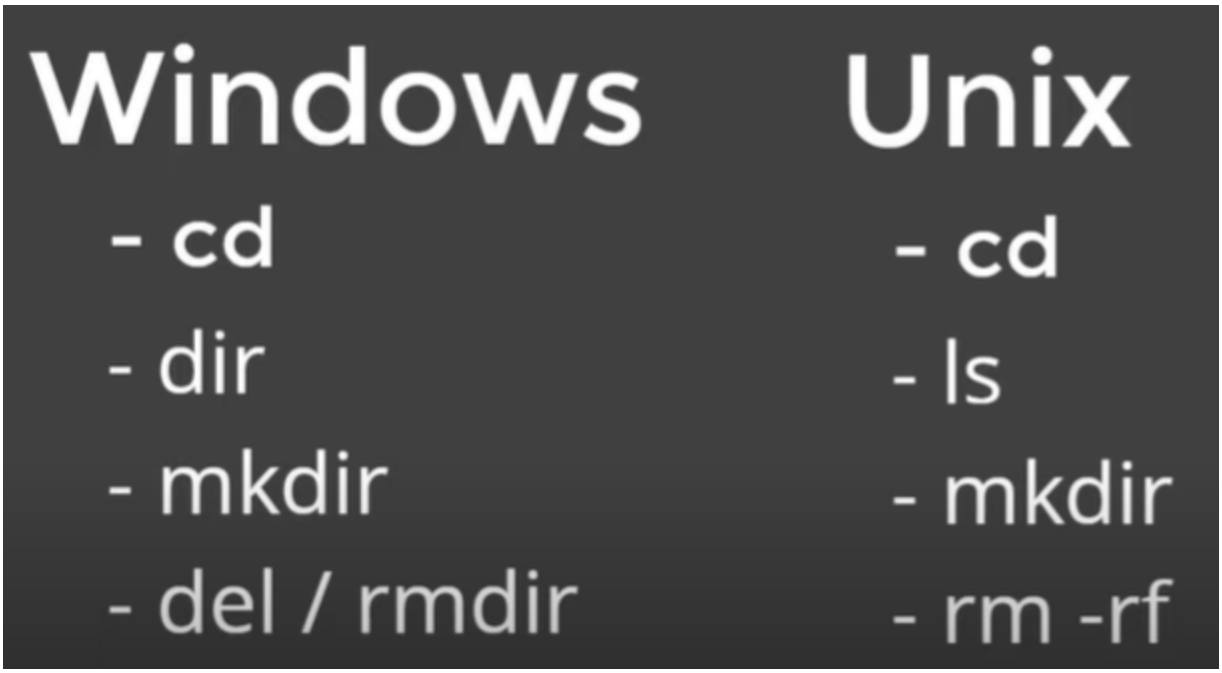
Objetos sao os vertices, nós do grafo.

Relacionamento sao arestas.

Estrutura de qualquer tipo, qualquer posição.

AULA 03 - GIT E GITHUB.

GUI X CLI(a forma de interagir é por linha de comando)



Windows = derivado do shell

Linux = do bash

⇒LISTAR

DIR para Win

LS para Linux

⇒Navegar nas pastas

CD / = volta para o C:

cd + Pasta (entra na pasta)

cd.. (retroceder um nível na navegação)

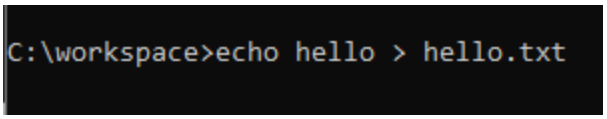
⇒LIMPAR CONSOLE

cls

⇒CRIAR PASTA

mkdir nome

⇒CRIANDO UM ARQUIVO

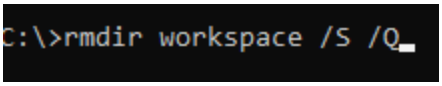


Vai pegar o imprimir e vai colocar dentro do arquivo.

⇒DELETAR ARQUIVOS

del = deleta arquivo e nao pasta.

⇒REMOVER PASTA/REPOSITORIO



flags = s e q.

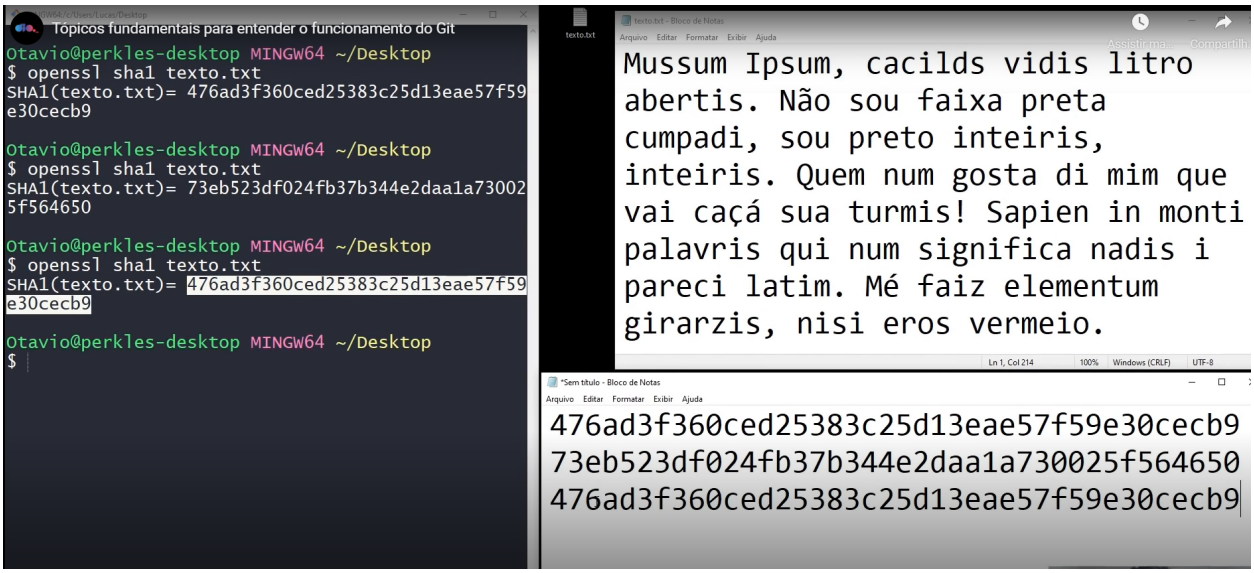
COMO O GIT FUNCIONA.

- SHA = algoritmo criptografado

Libera uma chave de 40 caracteres..

Arquivos que contenham os mesmos caracteres, ele gera o mesmo codigo.

Quando ele alterou o . para , gerou outra chave e quando voltou para o . voltou para a chave.



- OBJETOS DO GIT

⇒Blobs

onde armazena o objeto.

- - stdin = ele espera receber um arquivos, mas a gente esta enviando texto

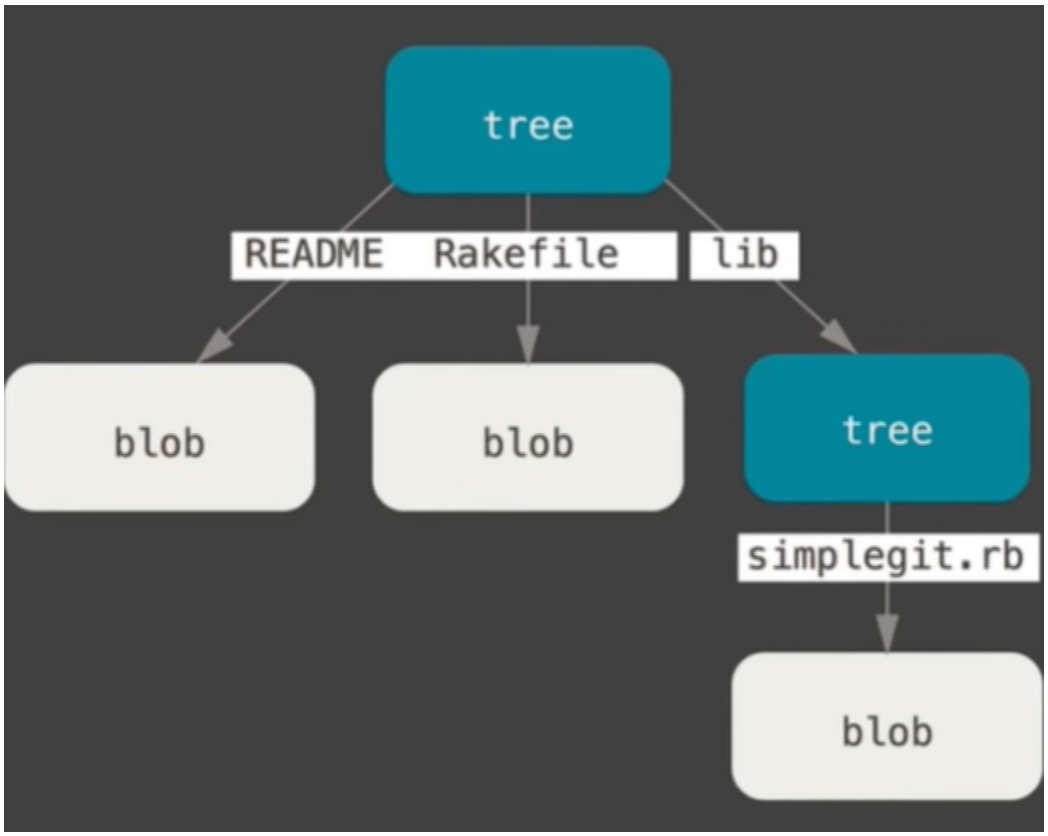
⇒Trees

Armazena e aponta para tipos de blobs diferentes ou para outras arvores.

Tambem tem um sha1

Contem tambem meta dados.

Também guarda o nome do arquivo



⇒Commits

Aponta para uma arvore, para um parente, para o autor e para uma mensagem.

Faz a junção de tudo.

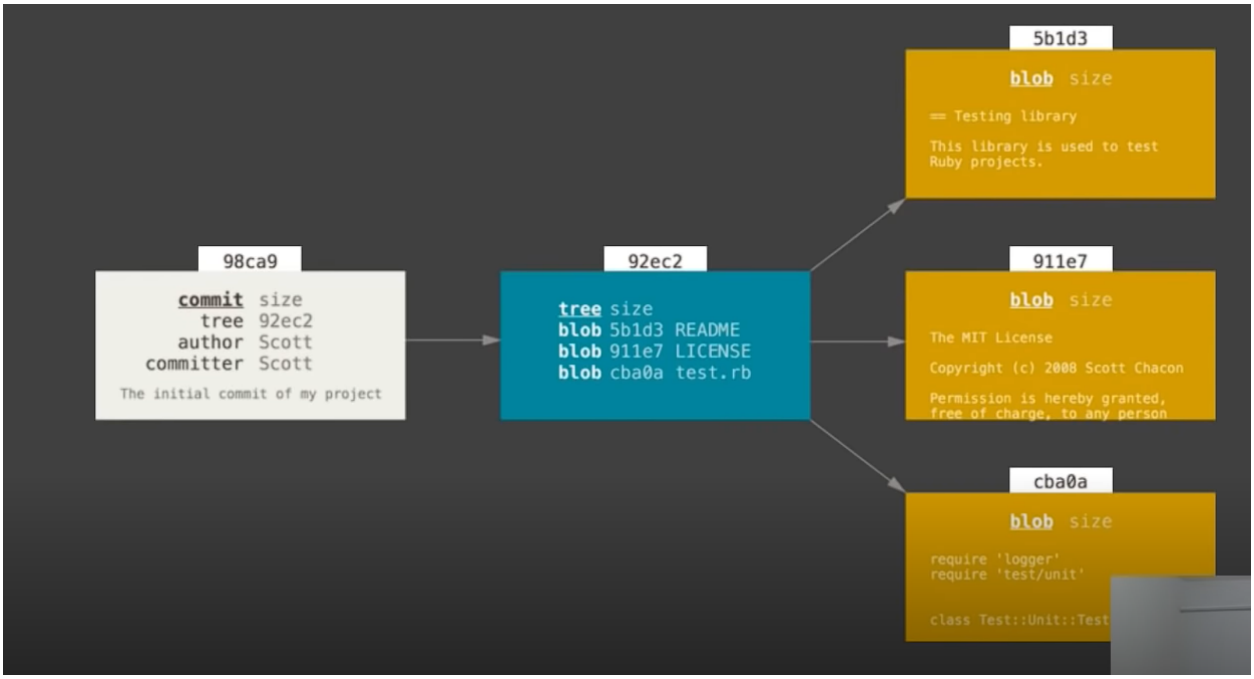
timestamp = carimbo de tempo (data, horario de criação)

Commits também possuem um sha1, hash de 40 caracteres.

Ou seja, voce alterar uma blob ele vai gerar uma sha1 daquela blob, essa blob por sua vez tem uma arvore apontando para ela, ou seja, vai alterar a arvore também e o commit aponta para uma arvores ou várias arvores. Qualquer coisa mudada, vai refletir em tudo.

Por isso o git é tao confiavel.

Monta uma linha do tempo.

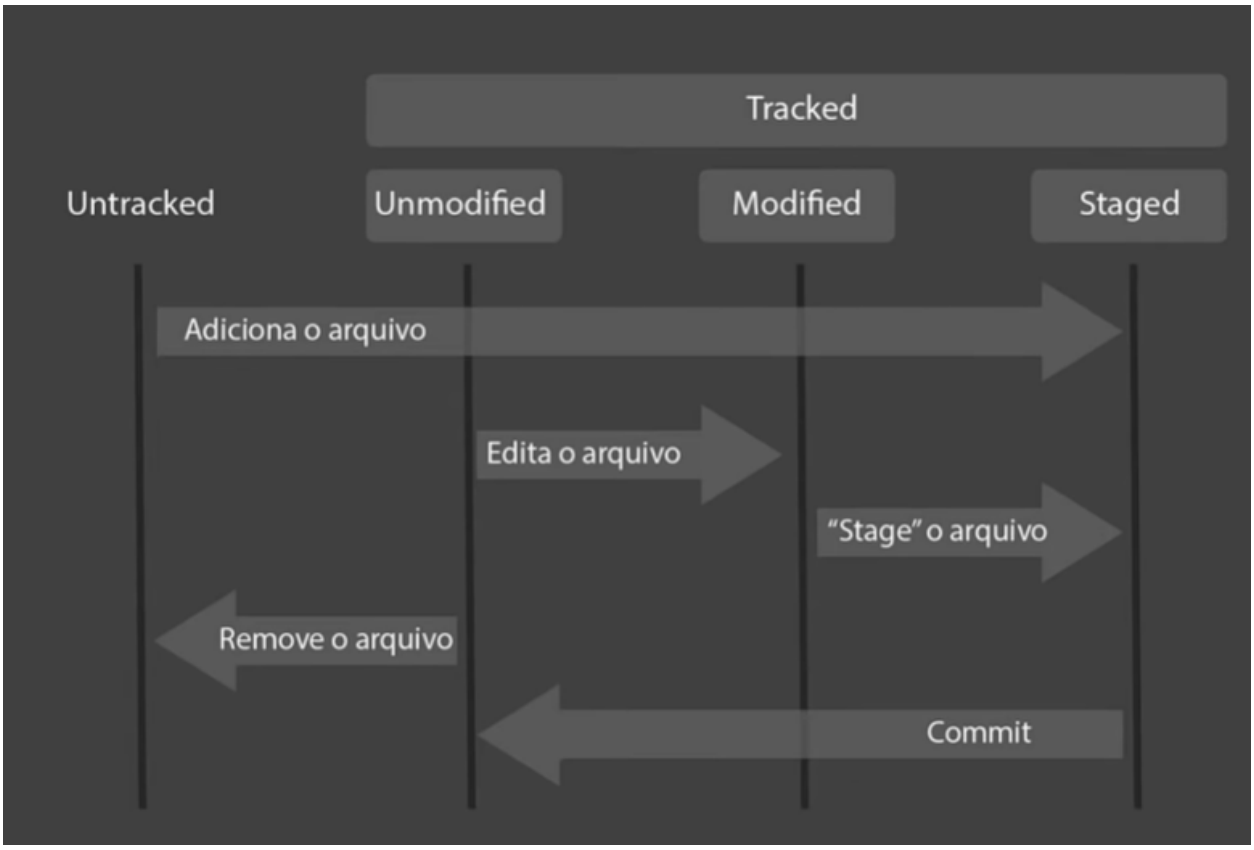


Obs: Ja criei uma chave privada e publica

username: lariguerra

usermail: sguerra.lariane@gmail.com

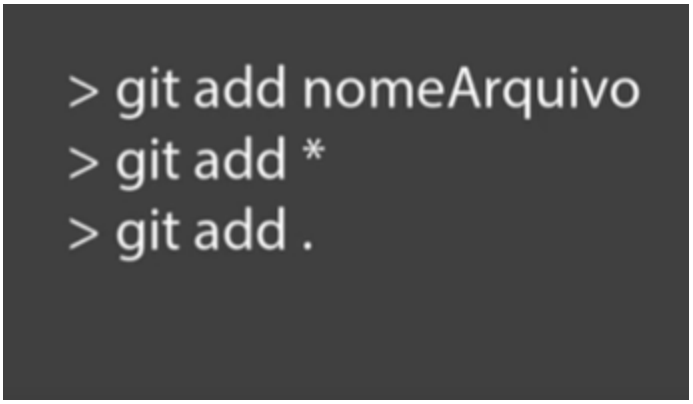
Mostrar ocultos:



git init ⇒ inicia um worksplace para virar um repositorio local

git status ⇒ descobre o status daquela works

git add * ou git add nome do arquivo (pode ser mais de um arquivo) ou git add .



git commit -m “string”

git config - - list = tras todas as configurações do git

Antes de empurrar:

→passar de onde ele esta saindo.

Conflito de merge: tem 2 alterações na mesma linha

Puxar: pull

git clone url

git remote -v = ver as versoes do repositorio.

git status → git add * → git status → git commit -m “ string” → git push