**Introdução ao Git e GitHub**

Git

- Software criado em 2005 pelo Linus Torvalds (criador do Linux)

- Sistema/software de versionamento de código distribuído

- Linux foi criado de maneira colaborativa e para isso Linus precisava um sistema que criasse versões para pessoas do mundo inteiro

- Nasceu do descontentamento de Linus com os sistemas de versionamento que já existiam

- Git e GitHub são tecnologias diferentes, porém complementares

GitHub

- Repositório online da Microsoft para guardar as versões

Vantagens de utilizar Git e GitHub

1. Controle de versão
2. Armazenamento em nuvem (GitHub)
3. Trabalho em equipe
4. Melhorar seu código
5. Reconhecimento

**Comandos básicos para um bom desempenho no terminal**

Git é um CLI (Command Line Interface, ou seja, Interface da Linha de Comando), onde o usuário interage diretamente com o sistema em um ambiente em texto, digitando comando do teclado em um prompt de comando.

Windows (shell) / Unix (bash)

dir ls 🡪 lista todas as pastas do diretório

cd cd 🡪 navega entre as pastas

cd .. cd .. 🡪 acessa a pasta anterior

cls clear 🡪 limpar o terminal

tab tab 🡪 auto completar nome de pastas

mkdir mkdir 🡪 cria pasta (completar com nome da pasta)

echo echo 🡪 printa na tela do terminal a frase seguinte

del 🡪 deleta arquivos dentro do repositório

rmdir rm -rf 🡪 deleta o repositório ([r] deleta todas as pastas de dentro do repositório e [f] não pergunta se quer deletar)

A sigla SHA significa 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 conjunto de caracteres identificador de 40 dígitos.

SHA1 nada mais é do que uma forma curta de representar um arquivo (com letras e números).

Blobs – informações do arquivo

Tree – apontam para outras árvores ou blobs

Commit – aponta para uma árvore, para o último commit, para o autor e para mensagem

Se houver uma alteração no Blob, o SHA será modificado em todos os níveis.

git init 🡪 inicializa o git no repositório

tracked 🡪 já foi iniciado pelo git init

unmodified 🡪 não modificado

modified 🡪 sofreu modificação

staged 🡪 (após add) preparado para fazer parte do commit (envelopar todas as modificações e carrega todas as informações)

commit 🡪 retorna todos os arquivos para unmodified

untracked 🡪 não foi iniciado pelo git init

ls –a 🡪 mostra arquivos ocultos

git status 🡪 apresenta o status dos arquivos

mkdir 🡪 cria uma pasta

mv 🡪 move um arquivo para uma pasta

echo > README.md 🡪 cria um arquivo com esse nome

git add \* 🡪 adiciona todos os arquivos para serem commitados

git commit –m 🡪 adiciona todo o repositório num repositório local e adiciona uma mensagem

git config --global 🡪 traz todas as configurações que se utiliza no git

git config –global – unset user.email 🡪 reseta o e-mail do git

git config –global – unset user.nickname 🡪 reseta o nickname

repositório remoto 🡪 GitHub

git remote add origin 🡪 cria um repositório remoto

git remote –v 🡪 lista arquivos do repositório remoto

git push 🡪 empurra do repositório local para remoto

**Estrutura de Dados e Algoritmo**

**Listas**

A diferença entre listas e arrays é a de que as listas possuem tamanho ajustável, enquanto arrays possuem tamanho fixo.

Lista Ligada

Na estrutura do tipo lista existem os nós onde cada um dos nós conhece o valor que está sendo armazenado em seu interior além de conhecer o elemento posterior a ele: por isso ela é chamada de “lista ligada”, pois os nós são amarrados com essa indicação de qual é o próximo nó.

Lista Duplamente Ligada

A grande diferença das listas duplamente ligadas para as listas ligadas é que elas são bidirecionais. Vimos que, naturalmente, não conseguimos “andar para trás” em listas ligadas, pois os nós de uma lista ligada sabem somente quem é o próximo elemento. Nas listas duplamente ligadas, os nós sabem quem é o próximo elemento e também quem é o elemento anterior, o que permite a navegação reversa.

**Pilhas (Stack)**

Uma pilha é uma estrutura de dados que serve como uma coleção de elementos, e permite o acesso a somente um item de dados armazenados.

O acesso aos itens de uma pilha é restrito – somente um item pode ser lido ou removido por vez.

Empty Stack -> Push (inserir 1) -> Push (inserir 2) -> Pop (retirar 2)

Tipos de Pilhas:

LIFO ou PEPS – A estrutura do tipo PILHA LIFO (Last in First Out) ou UEPS (Último que Entra Primeiro que Sai), apresenta o seguinte critério: o primeiro elemento a ser retirado é o último que tiver sido inserido.

FIFO ou UEPS – A estrutura do tipo PILHA FIFO (First in First Out) ou PEPS (Primeiro que Entra Primeiro que SAI), apresenta o seguinte critério: o primeiro elemento a ser retirado é o primeiro que tiver sido inserido.

Enqueue (inserir 1) -> Enqueue (inserir 2) -> Enqueue (inserir 3) -> Dequeue (retirar 1)

**Filas**

A estrutura do tipo Fila admite remoção e inserção de novos de elementos, sujeita à seguinte regra de operação: o elemento removido é o que está na estrutura há mais tempo, ou seja, o primeiro objeto inserido na fila é também o primeiro a ser removido seguindo o conceito FIFO.

**Árvores**

É uma estrutura de dados que organiza seus elementos de forma hierárquica, onde existe um elemento que fica no topo da árvore, chamado de raiz e existem os elementos subordinados a ele, que são chamados de nós ou folhas.

**Tabela Hash ou de Espalhamento**

Uma tabela hash, de dispersão ou espalhamento é uma estrutura de dados especial, que associa chaves de pesquisa a valores.

Uma tabela hash é uma generalização da ideia de array, porém utiliza uma função denominada Hashing para espalhar os elementos, fazendo com que os mesmos fiquem de forma não ordenada dentro do “array” que define a tabela.

Por que espalhar?

A tabela hash permite a associação de “valores” a “chaves”.

Valores: é a posição ou índice onde o elemento se encontra

Chave: parte da informação que compõe o elemento a ser manipulado

Espalhar facilita a busca na estrutura de dados, pois a partir de uma chave podemos acessar de forma rápida uma posição do “array”

**Grafos**

Grafos são estruturas que permitem programar a relação entre objetos

Os objetos são vértices ou “nós” do grafo

Os relacionamentos são arestas