**INTRODUÇÃO AO GIT/GITHUB**

**GIT**: é um sistema de versionamento de código distribuído, ou seja, nos ajuda a criar várias versões de nosso código para que possamos monitorar e fazer as manutenções.

**GITHUB:** armazena na infraestrutura de nuvem da Microsoft os seus códigos que foram versionados através do GIT, deixando publico para que outras pessoas possam ver e até mesmo contribuir.

**GUI (Grafic User Interface)**: interface gráfica do usuário onde o prório pode interar clicando, arrastando para executar uma tarefa.

**CLI (Command Line Interface):** interface com linhas de comandos onde o usuário interage através de comandos/códigos passados em um terminal para que seja executada uma tarefa.

**Comandos Básicos no terminal (Windows – Linux/Mac):**

1. **DIR – LS:** lista todas as pastas e arquivos para nós.
   1. Dir -a: mostra os arquivos ocultos
2. **CD:** muda de pasta em nosso diretório
   1. Cd / : vai direto para o diretório inicial.
   2. Cd pasta-desejada: muda para a pasta desejada.
   3. Cd .. : volta um nível no terminal.
   4. Cls – Clear: limpa a tela do terminal.
3. **TAB**: autocompleta o comando.
4. **Mkdir:** cria uma pasta/diretório.
   1. **Mkdir** pasta nova: cria uma nova pasta chamada pasta nova.
5. **Echo**: printa algo no terminal.
   1. Echo hello world: imprimi hello world no terminal.
6. **>** : é um redicionador de arquivo
   1. Echo hello > hello.txt : cria um arquivo chamado hello.txt que contém a saida da função echo.
7. **Del:** deleta arquivos de uma pasta.
   1. Del pasta x: deleta todos os arquivos da pasta x
8. **Seta para cima**: mostra o histórico de comandos que eu já realizei no terminal.
9. **Mv**: move um arquivo de uma pasta para outra
   1. Mv arquivo ./repositório destino
10. **Rmdir - Rm:** remove uma pasta com todos os arquivos que ela contém.
    1. Rmdir pasta y /S /Q: deleta a pasta y e todos os arquivos que ela tem.
    2. Rm -rf pasta z: deleta a pasta z e todos os aquivos que nela existe.

**Tópicos fundamentais para poder entender o GIT**

**SHA1 (Secure Hash Algorithm):** é um algoritmo de encriptação, que gera uma encriptação com quarenta caracteres. Se uma vírgula é mudada no arquivo, uma nova encriptação de caracteres é gerada.

1. Openssl shal arquivo : vai encriptar o arquivo gerando uma cadeia de 40 caracteres.

**Objetos internos do GIT:**

1. **BLOBS**: armazena os arquivos em formato de metadados. Uma Blob vai ter o tipo do objeto (uma blob), o tamanho dessa string/arquivo, uma \0 e o nome de fato do arquivo.
2. **TREES**: armazenam e apontam para os Blobs ou outras Trees. Uma Tree vai ter o objeto blob e/ou blob, o tamanho da string/arquivo, e o nome dos arquivos. Também possui um SHA1.
3. **COMMITS**: Concatena tudo. Um Commit aponta para uma Tree, aponta para um parent (um Commit realizado antes dele), um autor e uma mensagem. Também possuem um SHA1.



**Chave SSH:** é uma forma de estabelecer uma concexão segura e encriptada entre duas máquinas. Como gerar uma chave SSH:

1. $ ssh-keygen -t ed25519 -C [meuemail@email.com](mailto:meuemail@email.com)
2. Vá até a pasta em que as chaves foram geradas.
3. $ cat id\_25519.pub
4. Copie o output e cole no Github
5. Agora vamos iniciar o processo para poder usar a chave em nossa máquina para acessar o github diréto da nossa máquina.
6. $ eval $(ssh-agent -s)
7. Agora vamos passar a nossa chave privada para o agente
8. $ ssh-add chave\_privada

**Primeiros comandos com o Git**

Todos os comandos do git vem com o nome do git primeiro.

1. **Git init:** cria repositórios (uma pasta)
2. **Git add:** dá inicio ao versionamento do código adicionando os arquivos
3. **Git commit:** dá o commit
4. **Git config:** permite você confirar certas opções em seu commit
5. **Git status:** nos diz em qual etapa o nosso arquivo se encontra (unmodified, modified, staged)
6. **Git remote add:** adiciona nosso repositório local ao remoto no GitHub
   1. Git remote add origin “https do repositório remoto no github”
   2. Git remove -v: lista os repositórios cadastrados
7. **Git push**: empurra nosso repositório local para o remoto

**Ciclo de vida dos arquivos dentro do GIT**



**Untracked:** são os arquivos que o Git não tem conhecimento sobre (git init)

**Tracked:** são os arquivos que o Git passa a conhecer (git add)

1. **Unmodified:** um arquivo que ainda não sofreu modificação
2. **Modified:** um arquivo que já sofreu modificação (muda o SHA1)
3. **Staged:** um arquivo que já está pronto para sofre o commit e retornar para unmodified, salvando o arquivo em nosso repositório local.