**Introdução ao Git e ao GitHub**

Ex: Jornalista: Rascunho 🡪 Pessoas 🡪 Reportagem

Rascunho pode virar o rascunhoantigo1, submeto de novo, faço rascunho, pode virar rascunhoantigo1, enquanto o que era rascunhoantigo1 vira rascunhoantigo2

**Git 🡪** Software que facilita isso de versão antiga, versão final, é um sistema de versionamento (ajuda a criar e monitorar diferentes versões) de código

Dica: softwares não são feitos sozinhos, mas sim colaborativamente

Benefícios aprender Git/GitHub: controle de versão, armazenamento em nuvem, trabalho em equipe, melhorar seu código, reconhecimento

GUI X CLI

Gui (graphic user interface) 🡪 programas com interface gráfica. Usuário clica, arrasta, seleciona, usa atalhos, é tudo responsivo

Git é CLI 🡪 Command Line Interface

Comandos CLI:

* **Dir**: Listar, se situar, entender em que local estamos. Traz uma lista de diretórios contidos na pasta que estamos situados
* **Cd** (change directory): Ir para uma pasta específica, possibilita navegar entre as pastas. Ex: **cd / 🡪** leva pra base

**cd Windows 🡪** leva pra pasta Windows

**cd .. 🡪** retrocede uma pasta, um nível

**cls 🡪** limpa a tela

**tab 🡪** auto completar

**mkdir 🡪** cria uma pasta, ex: mkdir workspace cria uma pasta com esse nome

**echo 🡪** simplesmente printa no terminal um texto que eu passe

**echo hello > hello.txt 🡪** cria um arquivo txt com o texto hello escrito

**del 🡪** del se restringe a deletar arquivos, ex: del workspace não vai deletar a pasta, só os arquivos dentro dela

**seta pra cima 🡪** navega pelo histórico de comandos que executei

**rdmir 🡪** remove diretório, ex: rmdir workspace /S /Q

ps:

/s Deletes a directory tree (the specified directory and all its subdirectories, including all files).

/q Specifies quiet mode. Does not prompt for confirmation when deleting a directory tree. The /q parameter works only if /s is also specified.

Como o Git funciona por debaixo dos panos? (SHA1, Objetos fundamentais, Sistema distribuído e segurança)

**SHA1 🡪** algoritmo de encriptação (secure hash algorithm). Pega arquivo e gera conjunto de caracteres identificador de 40 dígitos

* É uma forma curta de representar um arquivo
* Lá no git bash, se clico no desktop com o botão direito e inicializo, posso rodar um arquivo de texto com o nome texto com o sha1 digitando openssl sha1 texto.txt, ele vai gerar 40 caracteres. Se eu altero uma vírgula nesse arquivo, ele gera outra sequência de 40 caracteres, se volto o arquivo pra forma original, ele retorna a primeira sequência, que tinha retornado antes.
* É uma forma bem massa de garantir que os arquivos sofreram modificação e que de uma versão pra outra não tenha, por exemplo, sofrido modificações

**Objetos fundamentais (objetos internos do git) 🡪 blobs, trees e commits 🡪** 3 tipos básicos de objetos responsáveis pelo versionamento do nosso código

* **Blobs 🡪** blocos básicos de composição
* **Tree 🡪** armazena e aponta para tipos de blobs diferentes, ou outras árvores
* **Commit 🡪** mais importante, junta tudo, aponta pruma árvore, pro último commit realizado antes dele, pro autor e pra uma mensagem e uma timestamp

Por isso o git é um sistema distribuído seguro,

**Chave SSH –** Forma segura de autenticação

**Token de acesso pessoal –** Outra forma segura de autenticação

ghp\_CYeR2IWPkk9u47KNec9qoRFPWORDiA0qiVmv

Uso os dois pra clonar repositórios?

**PRIMEIROS COMANDOS GIT**

**Criando um repositório:**

**git init 🡪** inicia o respositório

**git commit 🡪** inicia commit, ex: git commit -m “commit inicial” 🡪 vai gerar um commit carregando a mensagem ‘commit inicial’

Ciclo de vida de um arquivo:

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

git status 🡪 ajuda a monitorar o estado do arquivo (unmodified, staged, etc)

git add 🡪 adiciona arquivos para a área de staged, pra depois serem comitados

* Upando o repo
* Fui lá no github, criei um repo, fui no bash:
* git remote add origin <https://github.com/cmalassise/livro-receitas.git>
* git remote -v
* git push origin master