**GIT**

Conceito: É um software de controle de versão (VCS).

* O versionamento é importante, porque sempre estamos lhe dando com muitas versões para a mesma coisa. Então, quando um código estiver bem estruturado, ao invés de zipar, damos o commit para os arquivos irem para o repositório local (que é o git). O que o git faz é ver o que se tem de código, verificar o commit mas recente e identificar a diferença de um para o outro, então, ao invés dele salvar 2 códigos grandes, ele só vai salvar a diferença e dizer exatamente onde aquilo está acontecendo (nº da linha de código).

Rectangle

Description automatically generated A picture containing icon

Description automatically generated A picture containing icon

Description automatically generated

Repositório local = GIT

**Push**

**Commit**

Repositório remote = GITHUB

* Vantagens:
  + Controle de histórico
  + Trabalho em equipe:

Rectangle

Description automatically generated

UX/UI

**Merge**

**Master**

* A picture containing icon

  Description automatically generatedRectangle

  Description automatically generated

Dev front

* Rectangle

  Description automatically generated

Dev back

* + Segurança
    - saber quem fez a alteraçã, quando e voltar para a versão que desejar (se desejado).
  + Organização
    - ramificação do projeto (front, back, desgner) para depois fazer o merge (junção dos códigos) > master (projeto principal);
      * Cada ramificação só pode mexer nos códigos da sua área.

**GITHUB**

Conteiro: É uma plataforma de rede social para programadores

Vantagens:

* Você pode colaborar com o código dos outros;
* Testar o código dos outros e dizer as issues, as necessidades, quais são os problemas
* Repositório ilimitado;
* Hospedagem de código-fonte;
* Características de rede social;
* GitHub pages integrado (projeto html);
* Colaboração;
* Forks (copia do projeto original de alguém para melhorias no seu próprio repositório, desde que siga as regras da licença)

Concorrentes do GitHub:

GitLab, Bitbucket, Phabricator, Gogs, Kallithea.

* **Aprendendo alguns comandos**

1. O comando pode mudar de a cordo com cada S.O.
2. Abrir o prompt de commando a partir do comando-> cmd no iniciar
3. Se o comando tiver certo, o prompt não vai te trazer nenhum erro nem nada
4. Atalhos
   1. Comandos para voltar 1 pasta “cd ..”
   2. TAB = tem função de auto completar

|  |  |  |
| --- | --- | --- |
| **Ação** | **Código windows** | **Código Linux / Apple** |
| Listar os diretórios do computador | **dir** | **$ ls** |
| Navegar entre as pastas e ir para a home | **cd /**  **dir** | **$ cd**  **$cd /nome da pasta** |
| Printa no terminal uma frase | **echo** + frase |  |
| Verificando se o caminho acessado é a home |  | **$ pwd** |
| Criando um diretório/pasta | **mkdir** | **$ mkdir** |
| Limpar a tela | **cls // ou //ctrl+L** | **$ clear** |
| Criando um arquivo | **echo** + palavra **>** nomeArquivo | **$ touch** |
| Flag que mostra arquivos ocultos | **ls -a** |  |
| Escrevendo algumas linhas dentro de um arquivo |  | **$ nano** |
| Listar o conteúdo |  | **$ cat** |
| Listar o conteúdo em ordem alfabética |  | **$ sort** |
| Mover arquivo ou renomear arquivo | **mv** | **$ mv** |
| Mostrando a diferença entre os arquivos |  | **$ diff** |
| Concatenando os arquivos em um novo arquivo |  | **$** **cat** com **>>** |
| Exibir a quantidade de linhas, caracteres e nomes |  | **$ wc -clw** |
| Remover pasta | **rmdir** | **$ rm -r** |
| Copiar arquivo |  | **$ cp** |
| Verificar o tamanho das pastas |  | **$ du -h** |
| Remover arquivos com extensão jpg | **del** | **$ rm -r** diretório**/\*.jpg** |
| Exibir a data atual do sistema |  | **$ date** |
| Exibir o nome do sistema operacional |  | **$ uname** |
| Salvando o histórico dos comandos em um arquivo e renomeando |  | **$ history >>** nomeArquivo**.txt** |
|  |  |  |

* Secure Hash Algorithm (SHA) é um conjunto de funções hash criptográficas projetadas pela Agência de Segurança Nacional dos EUA (NSA)
  + A encriptação gera um conjunto de caracteres identificador de 40 dígitos
  + Abrindo o git bash (com o botão direito, direto do desktop) e dando o comando: **openssl sha1** nomePasta**.txt**, será gerado uma hash
* Tipos de objetos do git responsável pelo versionamento do código
  + Blobs
    - Estrutura básica:
      * Tem o sha1 do arquivo
      * Os arquivos ficam no obj blob, então terá o tipo do obj (=blob), tamanho da string ou arquivo, \0 e o conteúdo do arquivo

Text

Description automatically generated

* + Trees
    - Possuem pastas
    - Armazena as blobs e aponta para blobs diferentes, seja outras trees (árvores) ou outros diretórios (pelo nome do arquivo) e possuem o sha1 dos metadados das árvores

Graphical user interface, diagram, application

Description automatically generated with medium confidence

* + Commits
    - Junta tudo
    - Aponta para:
      * uma árvore,
      * Um parente: o último commit realizado antes dele,
      * Autor
      * Mensagem
      * Timestamp: a data e hora de quando foi criado
    - O sha1 desse commit é o hash de toda essa cadeia

Graphical user interface, application

Description automatically generated

Qualquer modificação feita no arquivo vai interferir no sha1 da blob e de toda a cadeia, de modo que o sha de todas as partes envolvida, inclusive do commit será alterada, e por isso que o git é tão confiável.

Diagram

Description automatically generated

* Chaves SSH e Tokens
  + É uma forma segura de 2 máquinas se conectarem de forma critografada, ou seja, conectar o servidor do git hub a máquina local. Sempre tem 1 chave pública e 1 chave privada, sendo a chave pública a que fica no git hub.
    - Código para gerar as duas chaves a partir do git bush:
      * 1º comando: **ssh-keygen -t ed xxx -C** emailGitHub
      * 2º commando: +enter
      * 3º commando: colocar a senha + enter
      * 4º commando: colocar a senha + enter
      * 5º commando: **cd /c/Users/Mª Clara/.ssh**
      * 6º commando: **ls**
      * 7º commando: **cat id\_ xxx.pub** + enter
      * 8º commando: copier a chave pública e colocar no git hub
      * 9º comando - Verificar o SSH agent: **eval $(ssh-agent -s)**
        + É uma entidade que vai ficar encarregada de pegar as chaves e lhe dar com elas
      * 10º comando - Entregar a chave para descriptografar toda vez que chegar uma mensagem criptografada**: ssh-add id\_xxx**
      * 11º comando: Inserir a senha + enter
      * 12º comando - Clonar o repositório do git hub a partir do SSH – vai no git hub > code > ssh > copia o link
      * 13º comando - No git bush: **git clone +** paste o link
      * 14º comando: **Yes +** enter
* Primeiros comandos com o git:
  + Criar repositório git: **git init**
  + Mover arquivos e iniciar versionamento: **git add**
  + Criar um commit: **git commit**
* Conceitos de Tracked e untracked (rastreados ou não)
  + **Tracked**: são os arquivos que o git tem ciência dele e que uma vez entrado em tracked, ele entrará em um ciclo até que seja removido.
    - **Unmodified**: arquivos do git que não sofreram modificação, então, caso seja modificado, qualquer coisa, ele já irá diretamente para o modified, pois, o sha1 será comparado e irá detectar a alteração. Do contrário, caso o arquivo unmodified seja excluido, ele volta para o untracked e o git não terá ciência dele.
    - **Modified**: se rodar o git add mais uma vez, ele irá para o staged
    - **Staged**: os arquivos vão se preparando para fazer parte de outro tipo de agrupamento que é o commit, que por sua vez, envelopa todo o conteúdo e volta para o unmodified para o ciclo ser reiniciado.
  + **Untracked**: são os arquivos que o git não tem ciência dele, mas que se for dado o git add, ele sobe direto para o staged.

A screenshot of a computer

Description automatically generated with low confidence

* + **Passos:**
    - **Abre a pasta c:/workspace e o git bash**
    - **Cd workspace**
    - **Mkdir livro-receitas**
    - **Ls**
    - **Cd livro-receitas**
    - **Git init**
    - **Git config –global user.email “<email>”**
    - Git config --global user.name “<name>”
    - ////////////// Add o arquivo: Markdown
    - **Git add \***
    - **Git commit -m “commit inicial”**
* Tipos de arquivo /sintaxe para níveis do texto:

Markdown (app Typora), navegador e HTML

Graphical user interface, text, application

Description automatically generated

* Mesmo que a conexão do SSH seja estabelecida entre o servidor (repositório remoto = git hub) e a máquina local, mas as alterações que forem realizadas no ambiente de desenvolvimento, só subirão para o remote repository se for dado o commit para ir ao repositório local e após isso, forem executadas uma série de comandos, para dá um input no remote repository. Portanto, tudo que está no repositório local deve ser commitado, se não, não será possível subir para o repositório remoto.
  + O comando que verifica qual status está o arquivo (Tracked ou untracked) é o git status
  + Passo a passo para o commit
    - Abrir o git bush na pasta livro-receitas em workspace
    - **Dá um mkdir receitas**
    - **Mv strogonoff.md ./receitas/**
    - **Git status**
    - **Git add receitas/**
    - **Cd receitas/**
    - **Ls**
    - **Git add strogonoff.md**
    - **Git status**
    - **Cd ..**
    - **Git commit -m “**cria pasta, move arquivo para receitas**”**
    - **Ls**
    - **Echo > README.md**
    - **Ls**
    - **Git status**
    - No arquivo README.md modifica o conteúdo dele normal e volta para o git bush
    - **Git status**
    - **Git add \***

Text

Description automatically generated

* + - **Git commit -m “**adiciona index**”**

Chart, waterfall chart

Description automatically generated

* Trabalhando com o GitHub
  + É importante que o nome e o email que vc adotou no git seja igual ao do github e para conferir, vc vai no git bash e põe o comando **git config --list** e aí na lista você pode estar conferindo a linha que tem escrito “user.email=...” e “user.name”
  + Para reconfigurar é preciso que o usuário não tenha feito commit em nada para q o sha1 não seja alterado. Então, os comandos são:
    - **Q**
    - **Git config --global --unset user.email**
    - **Git config --global --unset user.name**
    - **Git config --list**
    - **Git config –global user.email “**<email>”
    - **Git config --global user.name** “<name>”
* Passando os arquivos de um repositório local para um repositório remoto:
  + Git remote add origin <https://github.com/MariaPaes/Livro-receitas.git>
  + Git remote -v
  + Git status
  + Git push origin master