Ambiente Dev

Desenvolver Algoritmos

Msc. Lucas G. F. Alves
e-mail: lgfalves@senacrs.com.br

Senac

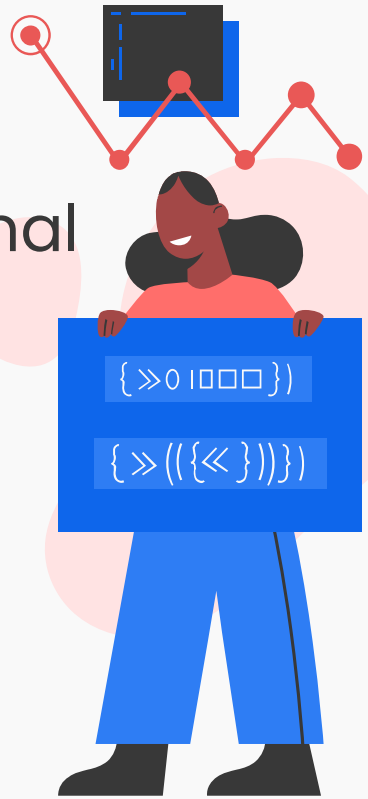
Planejamento de Aula

Revisão Pensamento Computacional

Introdução ao Computador

Comandos

Git



Introdução

Programação

Programar é dar instruções ao computador, como uma lista de tarefas do que deve ser realizado isso se chama algoritmo.

Para programar devemos pensar de forma lógica como o computador recebe os dados e os utiliza.

Para o computador receber essas instruções é utilizado uma linguagem de computador.

Pensamento Computacional

Algoritmo

Um algoritmo é uma sequência de passos ordenados necessários a um programa para que ele resolva determinado problema, como o de somar, subtrair, dividir, multiplicar números, procurar por palavras num texto, etc.

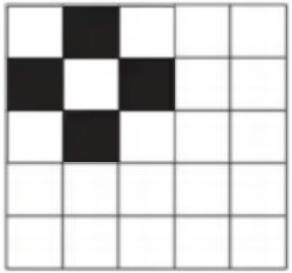
- É uma ordem ou sequência de passos para a solução de problemas.

Exemplo:

Receita de bolo.

Atividade em grupos – Algoritmo

Crie um desenho para outro grupo



Algoritmo

Mova para a direita, pinte o quadrado, mova para a direita, mova para baixo.

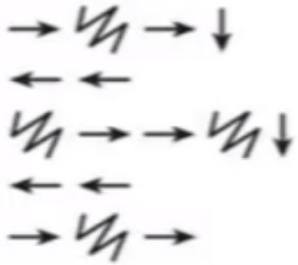
Mova para a esquerda, mova para a esquerda.

Pinte o quadrado, mova para a direita, mova para a direita, pinte o quadrado, mova para baixo.

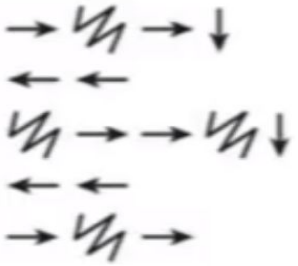
Mova para a esquerda, mova para a esquerda.

Mova para a direita, pinte o quadrado, mova para a direita.

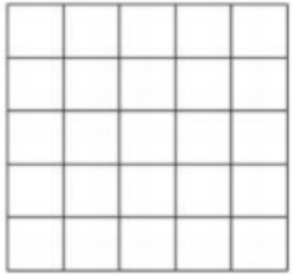
Código (forneça-o a outro grupo)



Código (fornecido pelo outro grupo)



Desenho obtido através do código fornecido



Códigos

Pensamento Computacional

Pensamento Lógico

Pensamento lógico é a habilidade de pensar de maneira estruturada e organizada, usando análise para resolver problemas de computador.

- É a habilidade de pensar como o computador.

Capacidade de dividir um problema em partes menores.

DIVIDIR PARA CONQUISTAR!

Pode ser desenvolvida com o tempo, prática e desafios.

Pensamento Computacional

Exemplo de algoritmo simples.

Na computação, uma soma de dois números é o exemplo mais clássico de um algoritmo simples. Primeiro é necessário ter uma sequência lógica de ações que envolvem os três elementos: entrada de dados, processamento e saída de dados.

Exemplo:

1 – Inserir o primeiro número
2 – Inserir o segundo número
3 – Somar os dois valores
4 – Mostrar o resultado

Pensamento Computacional

Linguagem de computador

Linguagem de computador é o idioma que o computador fala, cada linguagem com suas próprias regras e usos. Algumas linguagens são mais comuns para criação de aplicações de computadores, outras linguagens para aplicativos de smartphones, ou para construir sites ou para controlar robos.

- É o idioma que o computador entende.

Exemplo:

Python, JavaScript, C#, Java, C++, C.

Introdução a Pseudo-Código

Pseudo-código

É uma forma de representar o código, sejam algoritmos, funções ou outros processos, usando uma combinação de linguagem natural e elementos de programação.

Exemplos:

- Portugol.
- Scratch.

Introdução ao Computador

Introdução ao Computador

História do Computador

Alan Turing criou um modelo teórico de uma máquina que seria capaz de seguir instruções baseadas em 0 ou 1, realizando toda e qualquer operação computacional.

Turing também se envolveu na construção de máquinas físicas capazes de quebrar códigos de guerra alemães durante a segunda guerra mundial.

Filme: O jogo da imitação

Introdução ao Computador

Computadores

São máquinas poderosas capazes de:

- Cálculos complexos;
- Análise de dados;
- Jogos;
- Trabalho (textos, planilhas e apresentações);

Simples e intuitivo, focado em usuários "leigos".

Máquina que executa programas.

Introdução ao Computador

Sistemas Operacionais

São **programas** responsáveis por gerenciar o computador.

Organiza **vários programas** rodando ao mesmo tempo.

Gerencia o **sistema de arquivos** e pastas.

Apresenta **interface** para o computador, o que permite interação do usuário.

Introdução ao Computador

Interface

A interface é a ponte que faz a interação do usuário com o dispositivo ou programa.

Todos os comandos eram por texto, por meio de um Terminal **CLI - Command Line Interface** (Interface de linha de comando).

Com o tempo, foram desenvolvidas interfaces mais amigáveis e intuitivas, chamadas de **GUI - Graphical User Interface** (Interface Gráfica).

Temos também a **API - Application Programming Interface** (Interface de Programação), que permite a comunicação de diversos programas entre si.

Introdução ao Computador

Terminal

Programa que permite interação com o computador por meio de linhas de comandos (CLI).

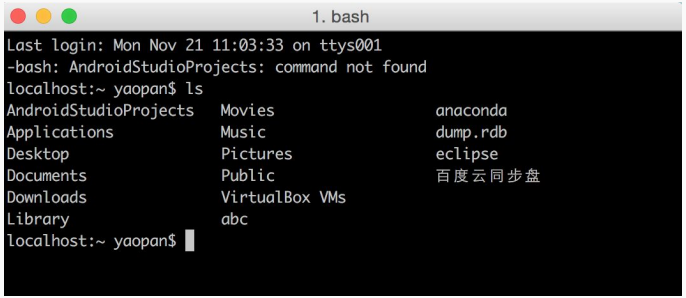
Existem vários terminais, alguns exemplos abaixo:

- Bash
- Zsh
- CMD
- PowerShell
- GitBash - Acessar comandos Unix

Introdução ao Computador

Terminal

Os terminais dependem do Sistema Operacional.

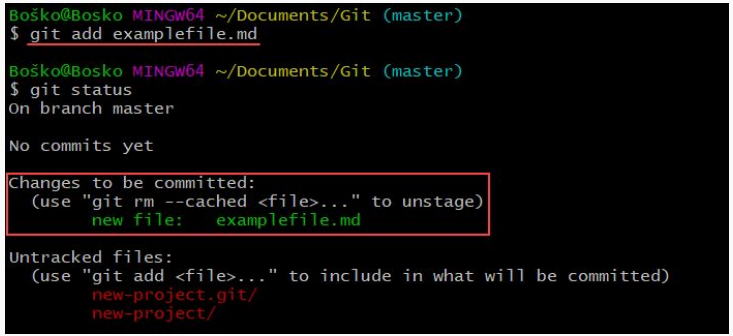


Sistemas Unix (Linux e MacOS)

- Herdaram o terminal da época em que não existiam interfaces gráficas.
- **Bash/Zsh**: um terminal muito poderoso.

Windows

- Foi desenvolvido com foco na interface gráfica.
- Usaremos, então, o **GitBash** que simula o bash no Windows.



Comandos

Comandos

O que são comandos?

Um comando é uma sequência de palavras e letras que executam uma determinada ação.

Cada elemento dessa sequência é chamado de **argumento**.

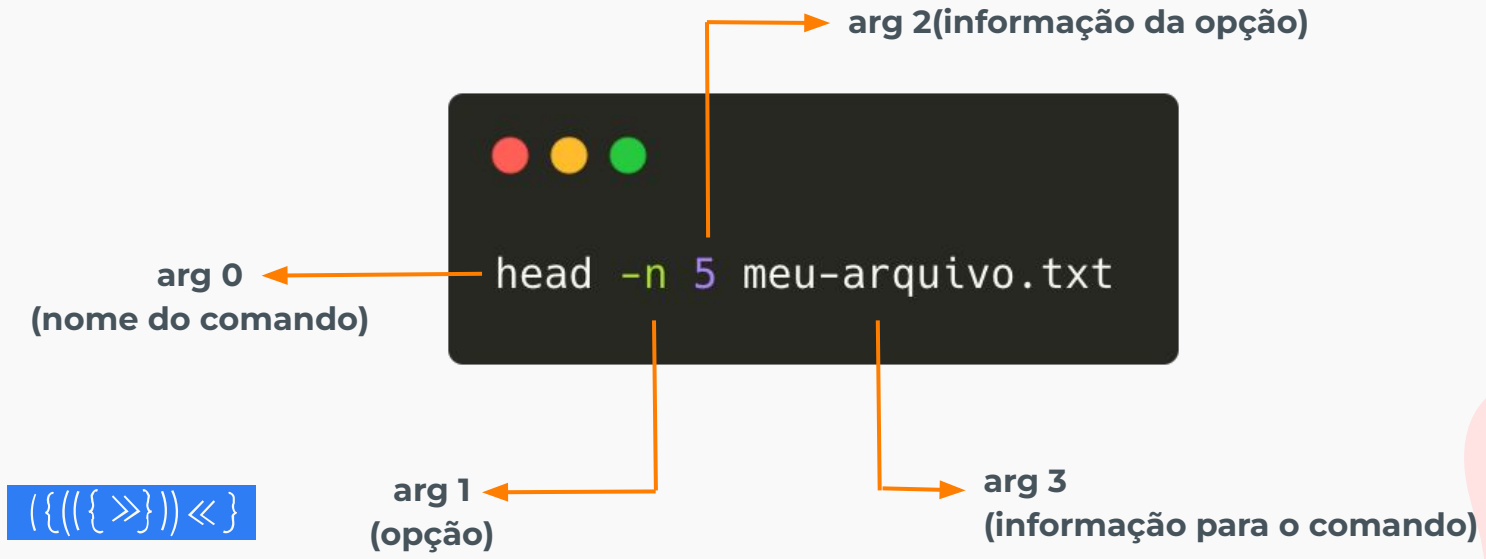
Cada comando pode possuir **opções** e **parâmetros.**

- **opções**: mudam o comportamento do comando e possuem o caractere “-” ou “--” como prefixo;
- **parâmetros**: são informações atribuídas ao próprio comando ou às opções;

Comandos

O que são comandos?

Exemplo: Visualizar as 5 primeiras linhas de um arquivo chamado arquivo.txt.



Comandos

echo

Imprime algo no terminal.

```
echo "Hello World" # imprime Hello World no terminal
```

whoami

Imprime o nome do usuário na tela

```
whoami # retorna o nome do usuário atual ex: 'seuUsuario'
```

Comandos

clear

Limpa tudo que está aparecendo no terminal, serve para leitura e organização.

```
clear #limpa tudo que está no terminal
```

pwd

Sigla para print working directory. Mostra o endereço completo do diretório/pasta em que estamos trabalhando.

```
pwd # retorna a pasta que o terminal está atualmente ex: '/c/Users/seuUsuario'
```

Comandos

ls

O nome vem de list. Lista os arquivos e pastas do diretório em que estamos.

```
ls # retorna nome de arquivos e pastas presentes na pasta atual
ls -a # retorna nome de arquivos e pastas, incluindo os ocultos (cujo nome começa com `.`)
ls -l # retorna informações de arquivos e pastas, incluindo tamanho, proprietário e outras informações
ls -la # soma dos dois modificadores anteriores
```

cd

Sigla para change directory. Muda o diretório/pasta em que estamos.

```
cd ./minha-pasta # troca o diretório atual para a subpasta "minha-pasta"
cd # vai para a pasta "home" do usuário atual
cd ../ # vai para a pasta acima da atual
```

Comandos

mkdir

Sigla para make directory. Cria um novo diretório.

```
mkdir minha-pasta # cria uma pasta chamada 'minha-pasta' no diretório atual
```

touch

Cria um novo arquivo.

```
touch index.html # criará um arquivo chamado index.html na pasta atual
```

Comandos

rm

Vem da palavra remove. Apaga arquivos de maneira IRREVERSÍVEL e SEM PEDIR CONFIRMAÇÃO.

```
rm ./meu-arquivo-gigante.txt # remove imediatamente o arquivo 'meu-arquivo-gigante.txt'
rm -r ./minha-pasta # remove Recursivamente todos os arquivos e sub-pastas da 'minha-pasta'
```

mv

Vem da palavra move. Permite mover ou renomear arquivos de um diretório.

```
mv ./meu-arquivo-gigante.txt ./minha-sub-pasta # move 'meu-arquivo-gigante' para 'minha-sub-pasta'
mv ./meu-arquivo-gigante.txt ./meu-gigante.txt # renomeia 'meu-arquivo-gigante.txt' para 'meu-gigante.txt'
```

Comandos

cp

Vem da palavra copy. Copia arquivos de um diretório para outro.

```
cp ./meu-arquivo-gigante.txt ./minha-sub-pasta # copia 'meu-arquivo-gigante' para 'minha-sub-pasta'
```

cat

Vem da palavra concat. Ele concatena tudo que está no arquivo e imprime no terminal.

```
cat meu-arquivo-gigante.txt # imprime o conteúdo do arquivo 'meu-arquivo-gigante.txt'
```

Comandos

head

Imprime as 10 primeiras linhas de um arquivo. A opção **-n** permite indicar quantas linhas são.

```
head meu-arquivo-gigante.txt # imprime as 10 primeiras linhas do arquivo 'meu-arquivo-gigante.txt'
head -n 20 meu-arquivo-gigante.txt # imprime as 20 primeiras linhas do arquivo 'meu-arquivo-gigante.txt'
```

tail

Imprime as 10 últimas linhas de um arquivo. A opção -n permite indicar quantas linhas são.

```
tail meu-arquivo-gigante.txt # imprime as 10 últimas linhas do arquivo 'meu-arquivo-gigante.txt'
tail -n 20 meu-arquivo-gigante.txt # imprime as 20 últimas linhas do arquivo 'meu-arquivo-gigante.txt'
```

Comandos

grep

Permite buscar um determinado texto no conteúdo de um arquivo

- -A x imprime x linhas após o texto
- -B y imprime y linhas antes do texto

```
grep Future4 ./lista-de-empresas.txt # Busca pela palavra Future4 no arquivo lista-de-empresas.txt e imprime toda a linha encontrada
```

Exercícios

Exercícios

1) Abrir o terminal dentro da pasta do template. Dica: botão direito do mouse em qualquer parte dentro da pasta ou no vs code.
2) Ler o conteúdo do arquivo de texto pokemons.txt. Dica: comando ‘cat’
3) Descobrir qual é o número do Pikachu. Dica: comando ‘grep’.
4) Descobrir os dois pokémons que vêm antes do Pikachu. Dica: comando ‘grep’ com opção -B.
5) Descobrir os três pokémons que vêm depois do Pikachu. Dica: comando ‘grep’ com opção -A.
6) Mostrar apenas os pokémons da primeira geração (do 1 ao 151). Dica: comando ‘head’ com opção -n.
7) Mostrar apenas os 100 últimos pokémons da lista. Dica: comando ‘tail’ com opção -n.

Git

Git

O que é Git?

O git é uma ferramenta que permite fazer o gerenciamento de versão de nossos projetos (de programação ou de outros arquivos).

Facilita o trabalho colaborativo.

Fácil de manter o rastreamento de arquivos que são alterados por duas pessoas ao mesmo tempo.

Git

Git vs. Github

O **Git** é a ferramenta que gerencia as versões e colaborações em projetos.

O **Github** é um serviço cloud que permite armazenar os projetos.

Existem outros, como **Bitbucket** e **Gitlab**. Todos usam a mesma ferramenta, o **Git**.

Github

Repositórios

O projeto que está na nossa máquina é chamado de:

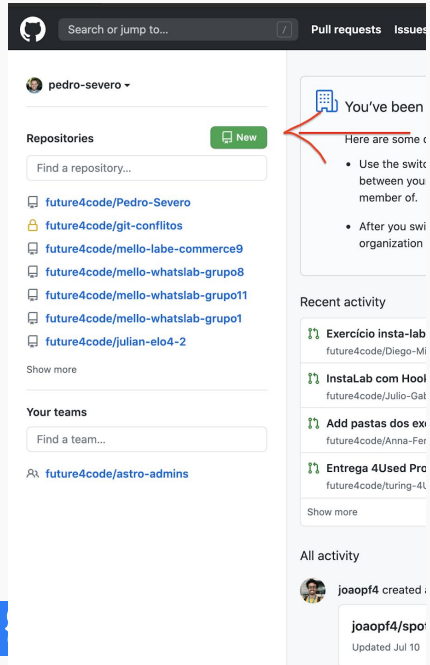
- **repositório (ou repo) do git local.**

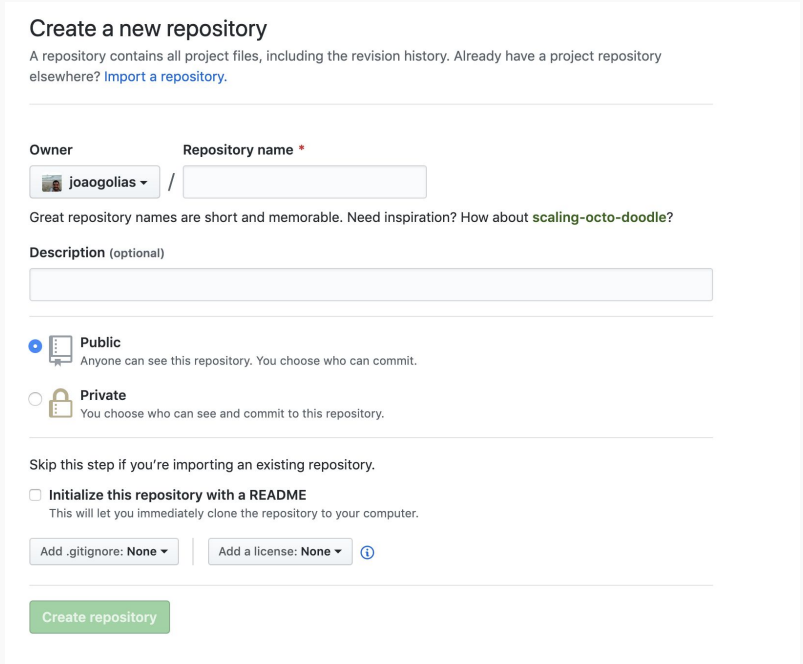
O projeto que está no github, chamados de:

- **repositório (ou repo) do git remoto.**

Github

Criando um repositório.

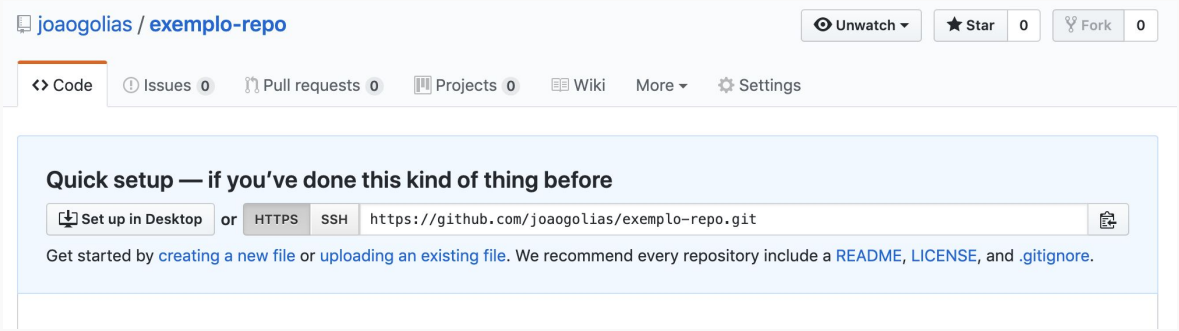




Github

git clone link-do-repo

É o comando que clona as informações do repositório remoto em uma pasta (repositório) na nossa máquina.



Github

Comandos do Github para salvar localmente

git status

Indica o status do repositório.

- Arquivos/pastas criados;
- Arquivos/pastas modificados;
- Arquivos/pastas removidos;

Github

Comandos do Github para salvar localmente

git add nome-do-arquivo

Envia os arquivos modificados, removidos e criados para a Staging Area (que é local).

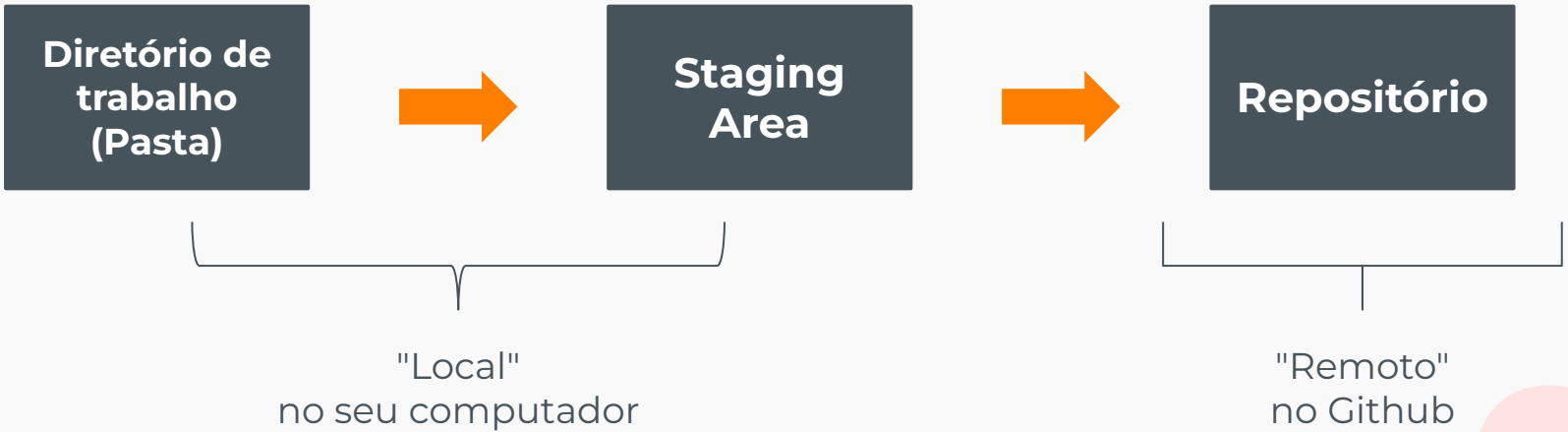
Também podemos utilizar a opção **git add --all** para adicionar todos os arquivos do repositório.

A opção **git add .** para adicionar todos os arquivos da pasta onde você se encontra.

Github

Comandos do Github para salvar localmente

git add .



Github

Comandos do Github para salvar localmente

git commit -m "mensagem"

Demarca uma versão do seu projeto com os arquivos que estiverem na Staging Area.

A mensagem deve explicar as modificações, criações e deleções feitas.

Github

Comandos do Github para salvar localmente

git commit -m "mensagem"

Demarca uma versão do seu projeto com os arquivos que estiverem na Staging Area.

A mensagem deve explicar as modificações, criações e deleções feitas.

Não esquecer do **-m**

Caso esqueça, você vai entrar em uma parte do terminal, que, para sair, você deve digitar: **esc esc :q**

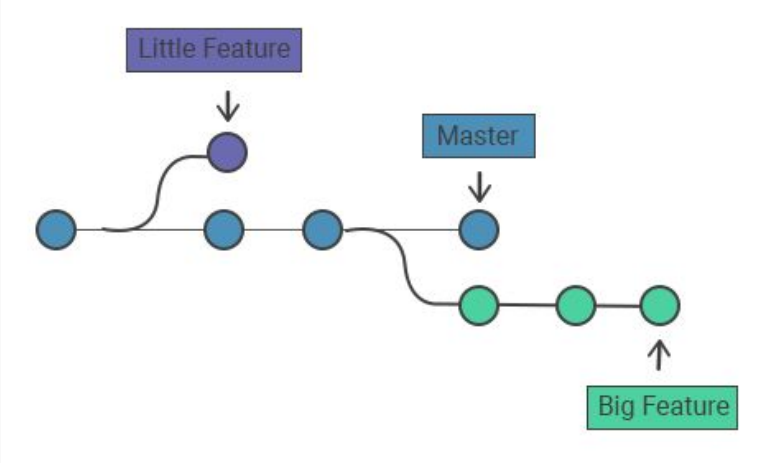
Não esquecer das aspas (").

Github

Dividindo o Trabalho

git branch

Branch (ramo/galho) é uma ramificação do projeto principal



Github

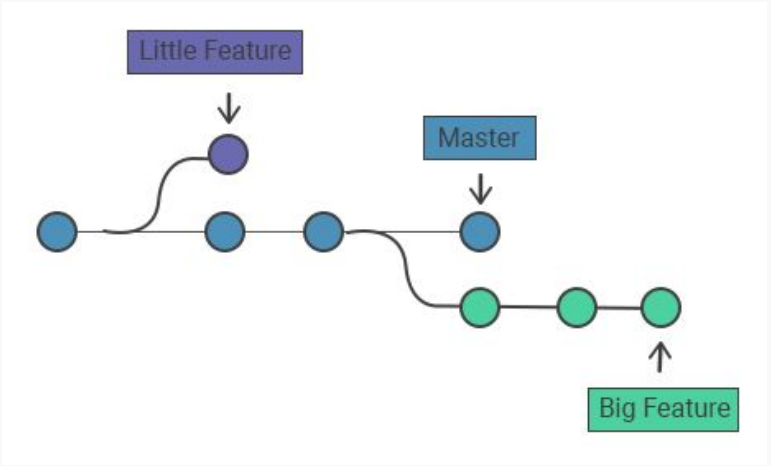
Dividindo o Trabalho

git branch

Branch (ramo/galho) é uma ramificação do projeto principal

Este comando em si mostra a lista de branches que estão no seu repositório local.

A branch padrão se chama main* e, a princípio, apenas ela vai existir no seu repositório.



Github

Dividindo o Trabalho

git branch nome-da-branch

Permite criar uma nova branch, com o nome que você escolheu.

git checkout nome-da-branch

Permite acessar uma branch que já foi criada (localmente ou remota).

git checkout -b nome-da-branch

É uma junção dos comandos anteriores.Ele cria uma nova branch e já acessa diretamente.

Github

Salvando no Remoto

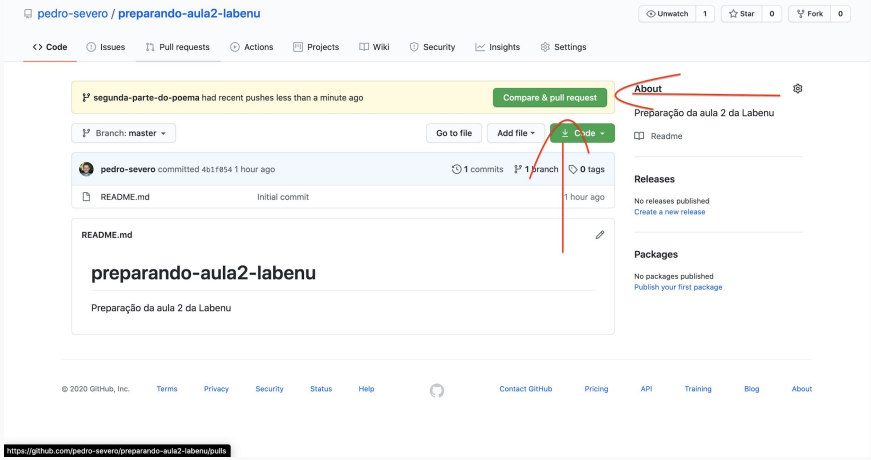
git push origin nome-da-branch

Envia as suas alterações feitas para a branch no repositório remoto Ele só envia as alterações que foram colocadas no commit

Github

Pull Request (PR)

Depois de fazer todas as alterações na sua branch, você deve querer que elas sejam mescladas com a branch principal (a master).
A esta **mesclagem**, damos o nome de **merge.**
Para fazer um merge no GitHub, nós devemos criar um Pull Request (ou PR) antes



Github

Pull Request (PR)

Quando trabalhamos em equipe, os membros dela avaliam os nossos PRs.

- Pedindo correções no código
- Sugerindo alterações

Após o processo de **Code Review** (CR) e o seu código estiver aprovado, ele pode ser **mergeado** na main.

Github

Atualizando o local

git pull origin nome-da-branch

Atualiza a branch em questão no seu repositório local com as alterações commitadas na branch remota.

Se você já estiver acessando a branch que deseja atualizar, o comando pode ser reduzido a **git pull**.

Exercícios

Exercícios

1) Criar um repositório chamado **DA-24-T.**
2) Clonar o repositório no computador local em C:Git.
3) Entrar na pasta do repositório utilizando o comando CD.
4) Criar a pasta aula 2 e colocar o arquivo pokemons.txt dentro da pasta.
5) Utilizar comando Git Status para verificar se apareceu o arquivo pokemons.txt.
6) Adicionar com o comando git add .
7) Commitar com o comando git commit -m “adicionado o arquivo pokemons.txt”
8) Salvar no repositório com o git push.

Github

Resumo

Começando o repositório

git clone link-do-repo

Salvando localmente

git status
git add nome-do-arquivo
git add .
git commit -m "mensagem"
git log

Github

Resumo

Dividindo o Trabalho

git branch
git branch nome-da-branch
git checkout nome-da-branch
git checkout -b nome-da-branch

Salvando no Remoto

git push origin nome-da-branch
git pull origin nome-da-branch

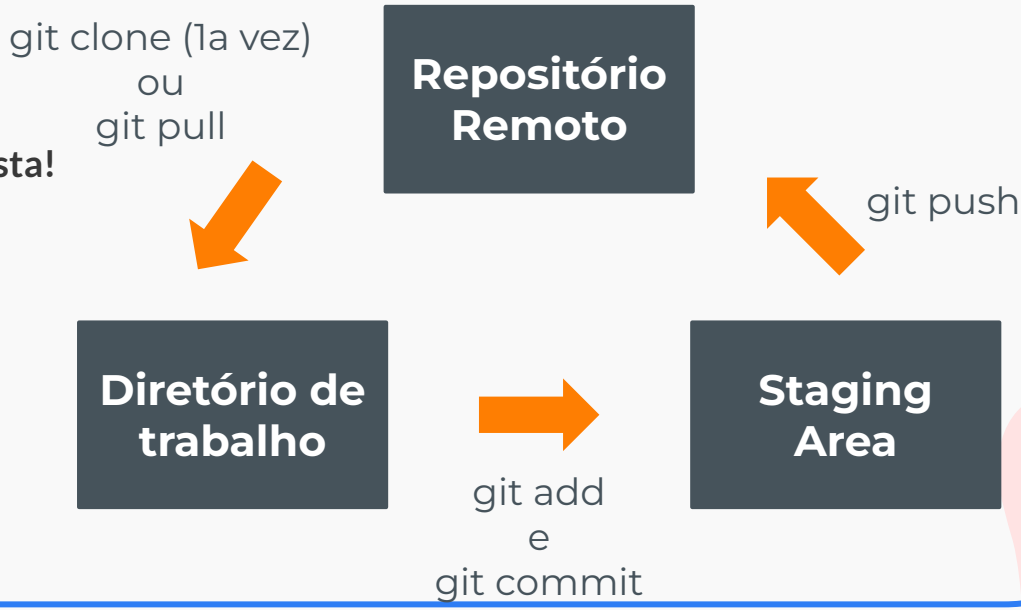
Github

Resumo

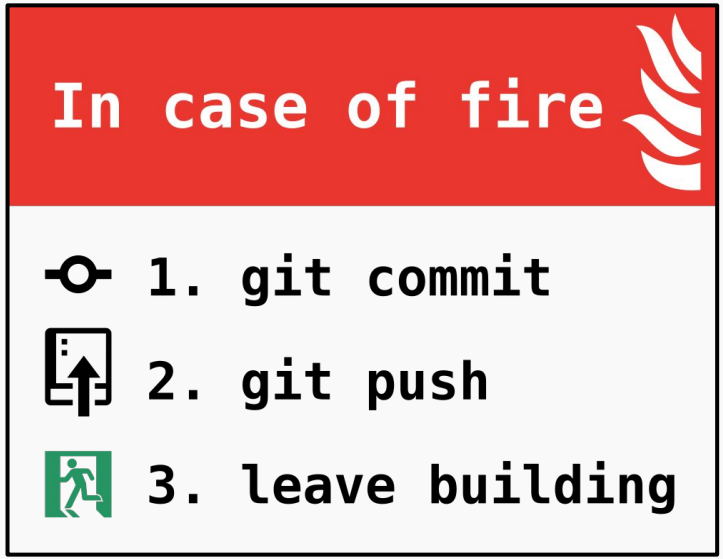
Importante: comandos de git **não são** o mesmo que comandos do terminal!

- Ex: git mkdir

Importante 2: **branch não é pasta!**



Github



Professor

Lucas G. F. Alves

Senac

Obrigado!

E-mail: lgfalves@senacrs.com.br