Introdução a Programação, Git e Github

Professora Amanda Silva

APRESENTAÇÃO

Cronograma

- Acordos e alinhamentos de expectativas
- Introdução a programação
- Comandos iniciais
- Versionamento, Git e GitHub
- Exercícios

Vou ser a sua professora

Amanda Silva

Acordos

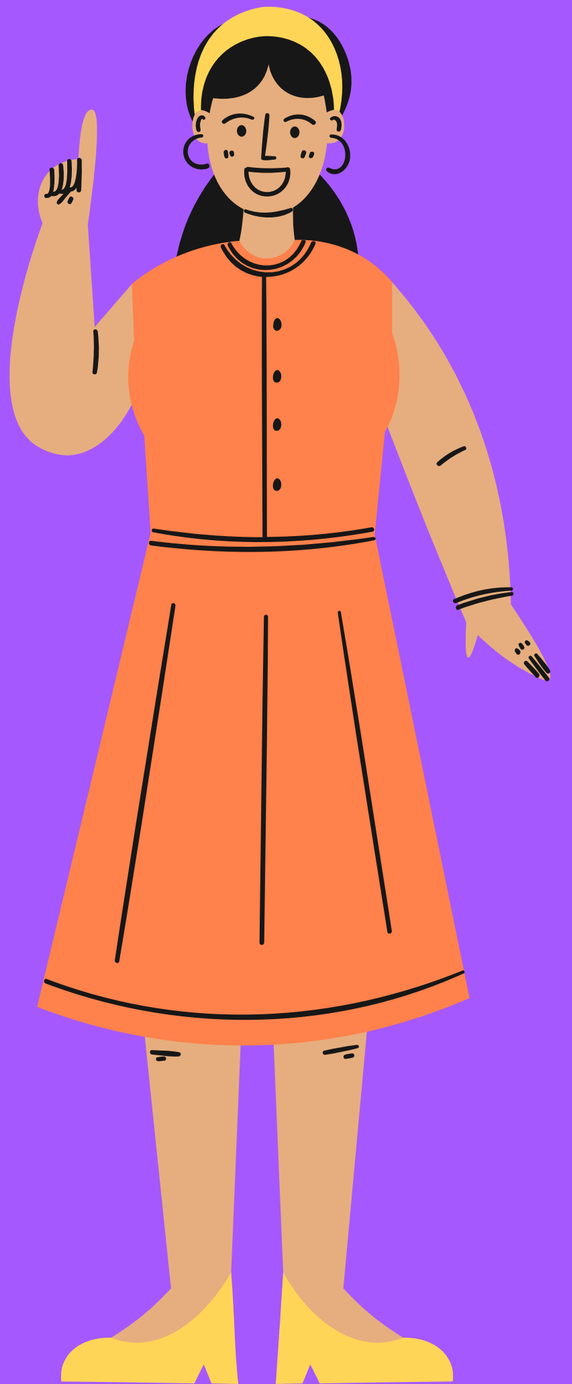
- Aqui é um ambiente seguro
- Não existe pergunta boba
- Pausas são importantes
- Feedbacks são tudo de bom
- Bebam água!

INTRODUÇÃO A PROGRAMAÇÃO

Hardware, software e programação

A **PARTE FÍSICA** DE UM COMPUTADOR É CHAMADA DE **HARDWARE** E POSSUÍ UMA LINGUAGEM COMPOSTA POR BITS, QUE SÃO ZEROS E UNS.

O **SOFTWARE** É O MEIO PELO QUAL A LINGUAGEM DE MÁQUINA PODE SER COMPILADA OU INTERPRETADA, ATRAVÉS DE CÓDIGOS CRIADOS EM UMA LINGUAGEM DE **PROGRAMAÇÃO**.

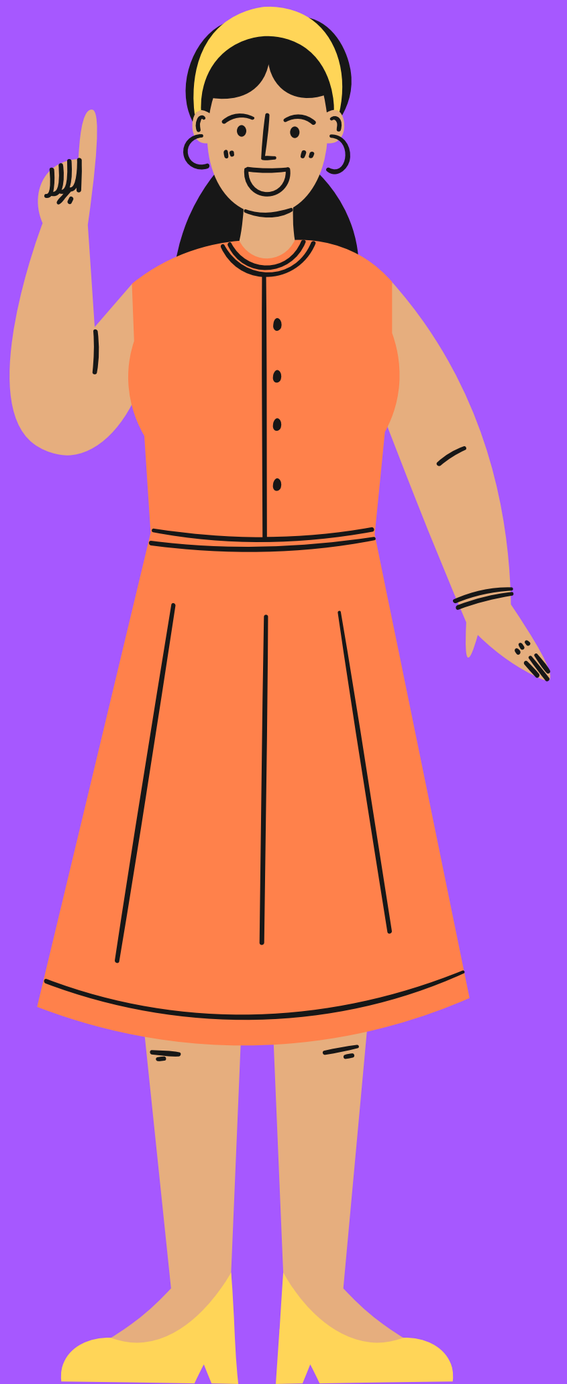


Hardware, software e programação

PROGRAMAÇÃO É EXATAMENTE O QUE POSSIBILITA A EXISTÊNCIA DE **SOFTWARES.**
POR CONSEQUÊNCIA, A UTILIZAÇÃO MAIS PRÁTICA DOS **HARDWARES**, JÁ QUE PERMITEM CRIAR PROGRAMAS QUE CONTROLAM O COMPORTAMENTO FÍSICO E LÓGICO DE UMA MÁQUINA.

Hardware, software e programação

ESSES PROGRAMAS, POR SUA VEZ, SÃO COMPOSTOS POR **CONJUNTOS DE INSTRUÇÕES** DETERMINADOS QUE **DESCREVEM TAREFAS A SEREM REALIZADAS** PELA MÁQUINA E ATENDEM DIVERSAS FINALIDADES, CHAMADOS DE **ALGORITMOS**.



Curiosidade:

O **PRIMEIRO** ALGORITMO DA HISTÓRIA FOI DESENVOLVIDO POR UMA **MULHER**, A **ADA LOVELACE**. ATRAVÉS DESSE ALGORITMO FOI POSSÍVEL A UTILIZAÇÃO DA MÁQUINA DE CHARLES BABBAGE, CONSIDERADA A **PRECURSORA** DOS COMPUTADORES ELETRÔNICOS ATUAIS.

MAS O QUE É ALGORITMO?

Mas o que é um algoritmo?

DE ACORDO COM O WIKIPÉDIA, UM **ALGORITMO** É UMA SEQUÊNCIA FINITA DE AÇÕES NARRATIVAS E EXECUTÁVEIS QUE VISAM OBTER UMA SOLUÇÃO PARA UM DETERMINADO TIPO DE PROBLEMA.

Sobre algoritmo:

O ALGORITMO PODE SER DIVIDIDO EM TRÊS ETAPAS: **ENTRADA**, **PROCESSAMENTO** E **SAÍDA**.

- ENTRADA RECEBE AS INFORMAÇÕES DA USUÁRIA PARA INICIAR O ALGORITMO;
- PROCESSAMENTO SÃO OS PASSOS NECESSÁRIOS PARA ATINGIR O OBJETIVO;
- SAÍDA É O RESULTADO FINAL ESPERADO DA FASE DE PROCESSAMENTO.

Lâmpada queimada:

1 - COLOQUE UMA ESCADA EMBAIXO DA LÂMPADA QUEIMADA.

2 - ESCOLHA UMA LÂMPADA NOVA DE MESMA POTÊNCIA/ VOLTAGEM DA QUEIMADA.

3 - SUBA NA ESCADA ATÉ ALCANÇAR A LÂMPADA QUEIMADA.

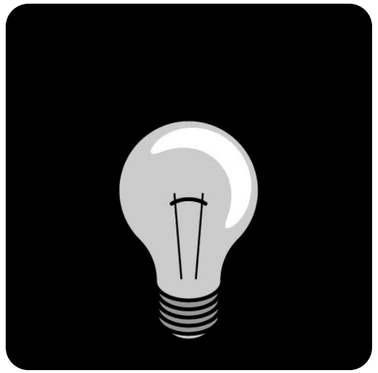
4 - GIRE A LÂMPADA QUEIMADA NO SENTIDO ANTI-HORÁRIO ATÉ QUE ELA SE SOLTE.

5 - POSICIONE A LÂMPADA NOVA NO SOQUETE.

6 - GIRE A LÂMPADA NO SENTIDO HORÁRIO, ATÉ QUE ELA SE FIRME.

7 - DESÇA DA ESCADA.

8 - GUARDE A ESCADA.



Outras formas de algoritmo

EXISTEM OUTRAS MANEIRAS DE ESCREVER UM ALGORITMO ALÉM DO FORMATO DE NARRATIVA. É POSSÍVEL CRIAR UM **FLUXOGRAMA** OU UM **PSEUDOCÓDIGO**.

Fluxograma

FLUXOGRAMA É UM **SISTEMA DE SÍMBOLOS E ABREVIAÇÕES** UTILIZADO PARA **REPRESENTAR UM PROCESSO** POR MEIO DE UM DESENHO, OU SEJA, O FLUXO DAS ATIVIDADES DE UM PROCESSO VISTO ATRAVÉS DE ÍCONES.

Início ou fim

Atividade a ser executada

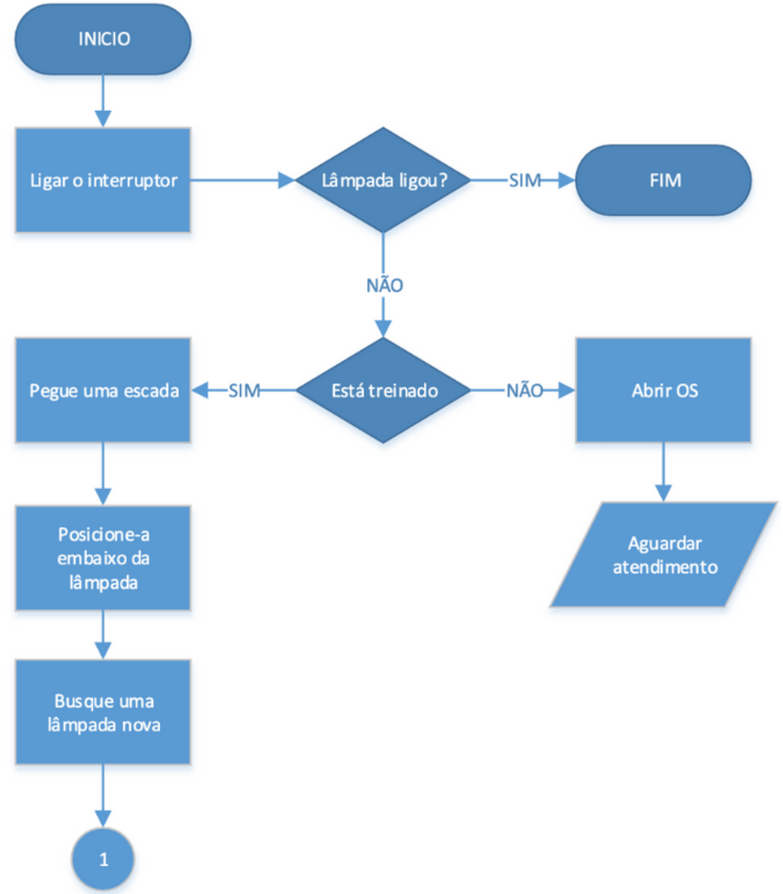
Ponto de tomada de decisão

Direção do fluxo

Documentos utilizados no processo

Espera

Fluxograma


INICIO
Ligar o interruptor
Lâmpada ligou?
SIM
FIM
NÃO
Pegue uma escada
SIM
Está treinado
NÃO
Abrir OS
Posicione-a embaixo da lâmpada
Aguardar atendimento
Busque uma lâmpada nova
1

Pseudocódigo

PSEUDOCÓDIGO É UMA FORMA DE **ESCREVER UM ALGORITMO**, COM UMA **LINGUAGEM SIMPLES** QUE PODE SER **ENTENDIDA POR QUALQUER PESSOA** SEM NECESSIDADE DE CONHECER ALGUMA LINGUAGEM DE PROGRAMAÇÃO.

Pseudocódigo (ou portugol)

TROCAR UMA LÂMPADA:

- PEGAR A ESCADA;
- POSICIONAR A ESCADA EMBAIXO DA LÂMPADA;
- BUSCAR UMA LÂMPADA NOVA;
- SUBIR NA ESCADA;
- RETIRAR LÂMPADA VELHA;
- COLOCAR LÂMPADA NOVA;

Linguagens de programação

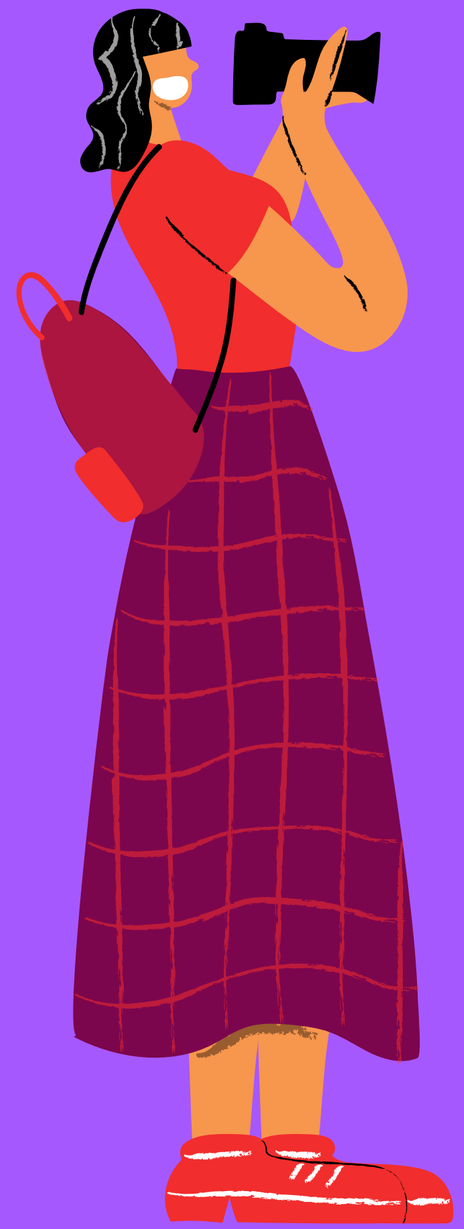
Linguagens de programação

TRATA-SE DE UMA **LINGUAGEM DE MÁQUINA** QUE ATRAVÉS DE **VÁRIAS INSTRUÇÕES**, PERMITE QUE UMA PESSOA ESCREVA UM CONJUNTO DE ORDENS, AÇÕES OU ATÉ RECEBA DADOS, **CRIANDO PROGRAMAS** QUE **CONTROLAM** O COMPORTAMENTO FÍSICO E LÓGICO DE UMA MÁQUINA.

Front-End

FRONT-END É CONSIDERADA A PARTE RESPONSÁVEL PELO **VISUAL** DE UM SITE OU APLICAÇÃO, OU SEJA, AQUILO QUE CONSEGUIMOS **VER** E **INTERAGIR**...

ALGUMAS LINGUAGENS DE BASE PARA O FRONT-END SÃO:
JAVASCRIPT, **TYPESCRIPT**, **HTML**(MARCAÇÃO) E **CSS**.
EXISTEM DIVERSAS BIBLIOTECAS E FRAMEWORKS TAMBÉM EM CONSTANTE EVOLUÇÃO.



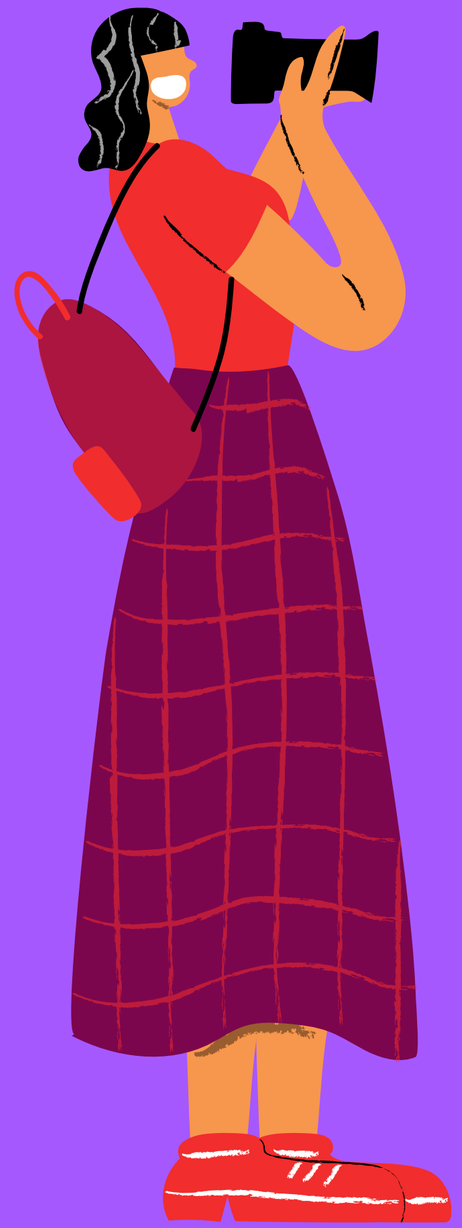
Back-end

BACK-END, REPRESENTA A PARTE POR TRÁS DE UMA APLICAÇÃO. É RESPONSÁVEL PELA OPERAÇÃO E DAS REGRAS DE NEGÓCIO E QUE NÃO INTERAGE COM A USUÁRIA DIRETAMENTE.
PODE TAMBÉM FAZER A PONTE ENTRE OS DADOS QUE VEM DO NAVEGADOR E O BANCO DE DADOS.

ALGUMAS DAS LINGUAGENS DO BACK-END:
PYTHON, **RUBY**, **JAVA**, **C#**, **JAVASCRIPT** (NODE.JS),
EXISTINDO TAMBÉM FRAMEWORKS E BIBLIOTECAS.

Banco de Dados

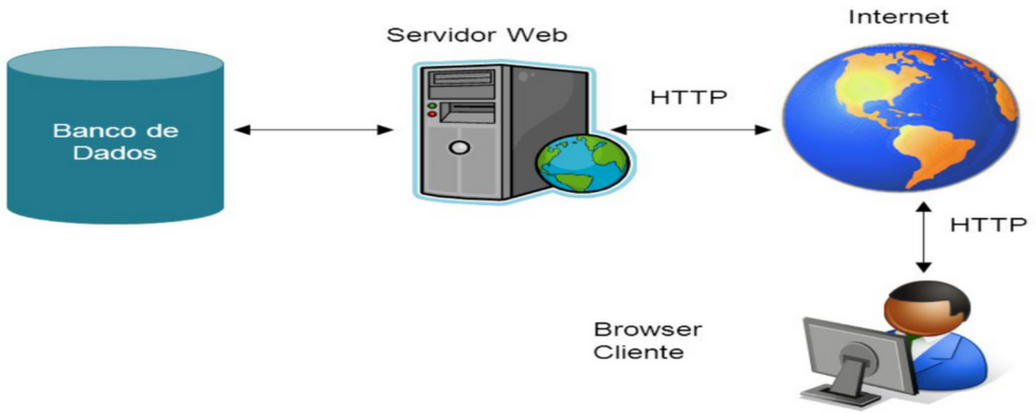
É UMA **COLEÇÃO DE DADOS** QUE SÃO ORGANIZADOS SOBRE UM DOMÍNIO ESPECÍFICO, EXPLICANDO DE MANEIRA SIMPLES, TRATA-SE DO **AGRUPAMENTO DE DADOS** QUE TRATAM DO MESMO ASSUNTO, E QUE PRECISAM SER **ARMAZENADOS** PARA SEGURANÇA OU CONFERÊNCIA FUTURA.



Cliente servidor

UMA APLICAÇÃO WEB É COMPOSTA POR DOIS ATORES PRINCIPAIS: **CLIENTE** E **SERVIDOR**.

O **CLIENTE** PODE SER UM NAVEGADOR. O **SERVIDOR** É UMA APLICAÇÃO, NA FORMA DE UM SERVIÇO, NORMALMENTE HOSPEDADO REMOTAMENTE.

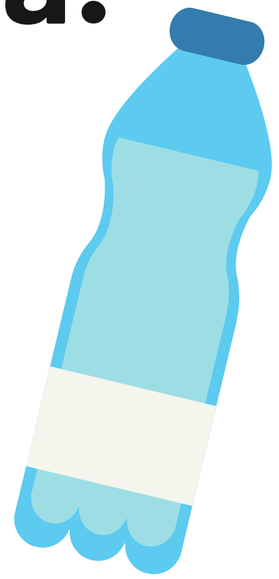


Você sabia?

HEDY LAMARR, FOI UMA ATRIZ DE HOLLYWOOD, QUE NA SEGUNDA GUERRA MUNDIAL **CRIOU UM SISTEMA** DE COMUNICAÇÕES PARA AS FORÇAS ARMADAS QUE MAIS TARDE SERVIU DE BASE PARA A CRIAÇÃO DO **WI-FI** E DA **TELEFONIA CELULAR**

Vamos fazer uma pausa rápida!

Voltamos em 10 minutos, e não se esqueçam de beber àgua



Linhas de Comando

Linhas de Comando

ALGUMAS INTERFACES INTERPRETAM LINHAS DE COMANDO ATRAVÉS DE TEXTOS QUE MANIPULAM ARQUIVOS EM NOSSOS COMPUTADORES.

- **COMMAND POWER / CMD**: INTERPRETADOR WINDOWS, SIMPLES E FUNCIONAL
- **POWERSHELL**: CRIADO PELA MICROSOFT, É BEM MAIS ROBUSTO COM CAPACIDADE MAIOR DE PROGRAMAÇÃO
- **BASH**: CRIADO COMO SOFTWARE LIVRE, É UM UNIX SHELL E LINGUAGEM DE COMANDO, ASSIM COMO O ANTERIOR

Comandos básicos

pwd : encontrar o caminho para o diretório atual (da pasta) em que você está
ls : listar todos arquivos da pasta que você está
cd nome-da-pasta : entrar em uma pasta dentro da pasta que você está
cd ~ : volta para a pasta raiz
cd .. : volta uma pasta para trás
mkdir nome-da-pasta : cria uma pasta
rm nome-do-arquivo : deleta um arquivo
rm -f ou **rm --recursive nome-da-pasta** : deleta uma pasta
whoami : identifica usuário que esta logado

Exercícios

Exercício 1:

- Abra o Bash
- Identifique o usuário
- ConfirmE a pasta que VOCÊ esta
- Crie uma pasta
- Entre na pasta
- Crie um arquivo e insira uma frase

Tire um print e mostra pra gente!

Exercício 1 - Resposta

- Abra o Bash
- Identifique o usuário: whoami
- Confirma a pasta em que esta: pwd
- Crie uma pasta: mkdir nome-da-pasta
- Entre na pasta: cd nome-da-pasta
- Crie um arquivo e insira uma frase

Pelo terminal:

- echo frase -> nome.txt cat -> nome.txt + enter e escreva o texto

Tire um print e mostra pra gente!

Exercício 2:

- Abra o terminal Bash
- Confirma a pasta em que você esta
- Entre na pasta criada antes
- Apague o arquivo criado
- Volte uma pasta
- Apague a pasta criada

Exercício 2 - Resposta

- Abra o Bash
- Confirma a pasta em que esta: pwd
- Entre na pasta criada antes: cd nome-da-pasta
- Apague o arquivo criado: rm nome-do-arquivo
- Volte uma pasta: cd ..
- Apague a pasta criada: rm -f nome-da-pasta

Hora do almoço

Voltamos às 13:30 horas, se alimentem, comam um docinho e voltem com aquela garrafinha de água

Versionamento de código

Controle de versão

Um software que **gerencia mudanças** de um arquivo de qualquer tipo(.doc, HTML, XML...), é muito usado nas empresas.
Através dele é possível **rastrear um arquivo** desde o início.
Podendo ser usado por qualquer tipo de projeto, desde os **mais simples** até os **mais complexos**, é utilizado na grande maioria das empresas.

Git

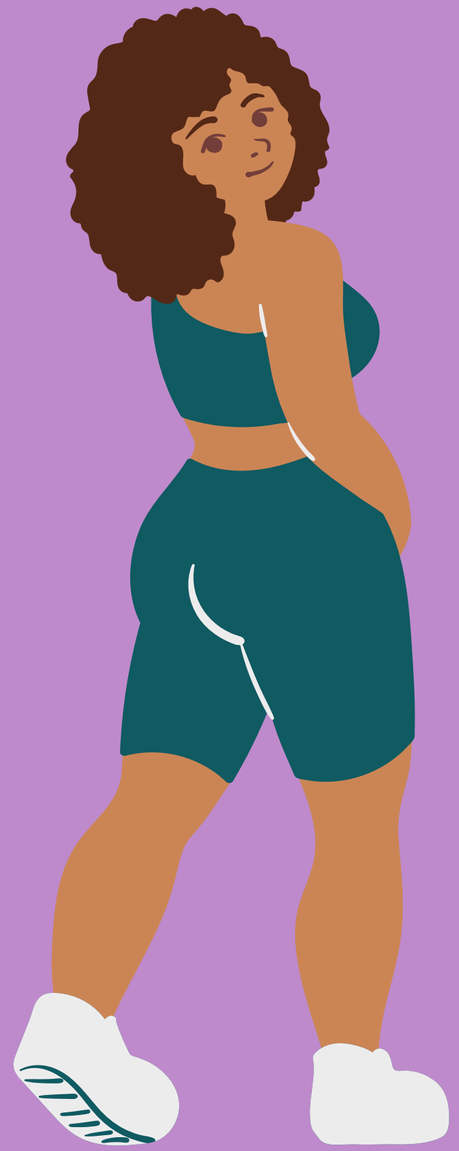
GIT É UM DOS **SISTEMA DE CONTROLE DE VERSÕES** EXISTENTES E O MAIS USADO, PRINCIPALMENTE NO DESENVOLVIMENTO DE SOFTWARE.

O GIT É UM **SOFTWARE LIVRE** E FOI INICIALMENTE PROJETADO E DESENVOLVIDO POR LINUS TORVALDS PARA O DESENVOLVIMENTO DO KERNEL LINUX.

ELE É INCRIVELMENTE **RÁPIDO**, É **MUITO EFICIENTE ,** INCLUSIVE EM PROJETOS GRANDES,

Git, comandos básicos

git init :
inicializa o git no repositório local
git add :
adiciona um arquivo modificado ao stagging (área temporária)
git status :
mostra os status dos arquivos modificados
git commit -m "mensagem" :
cria um commit



Git, comandos básicos

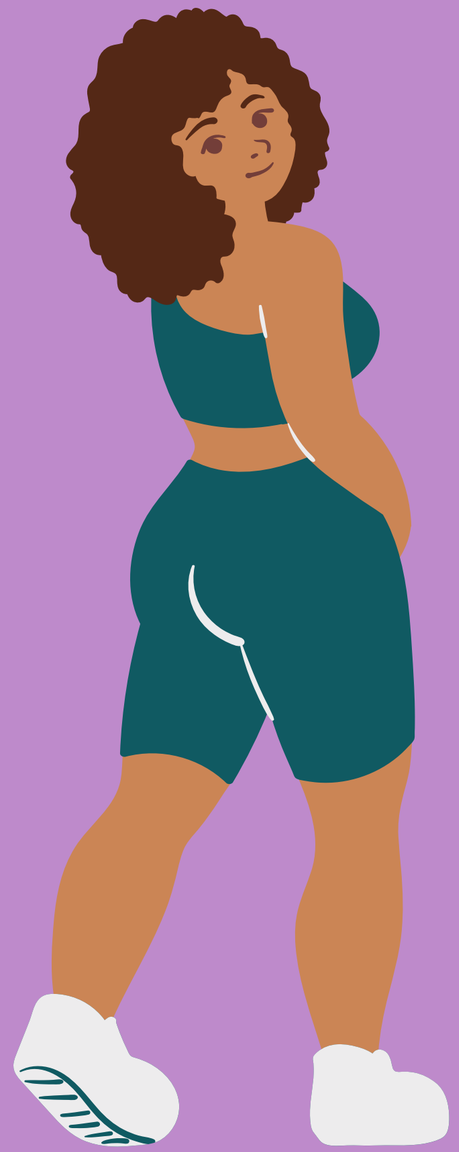
git pull :
puxa as atualizações mais recente (remoto -> local)
git push :
envia as atualizações mais recentes (local -> remoto)
git remote add origin caminho :
adiciona o seu repositório local ao remoto
git checkout -- nome-arquivo : descarta as alterações locais do arquivo informado

Ferramentas de versionamento

GitLab

Bitbucket

GitHub

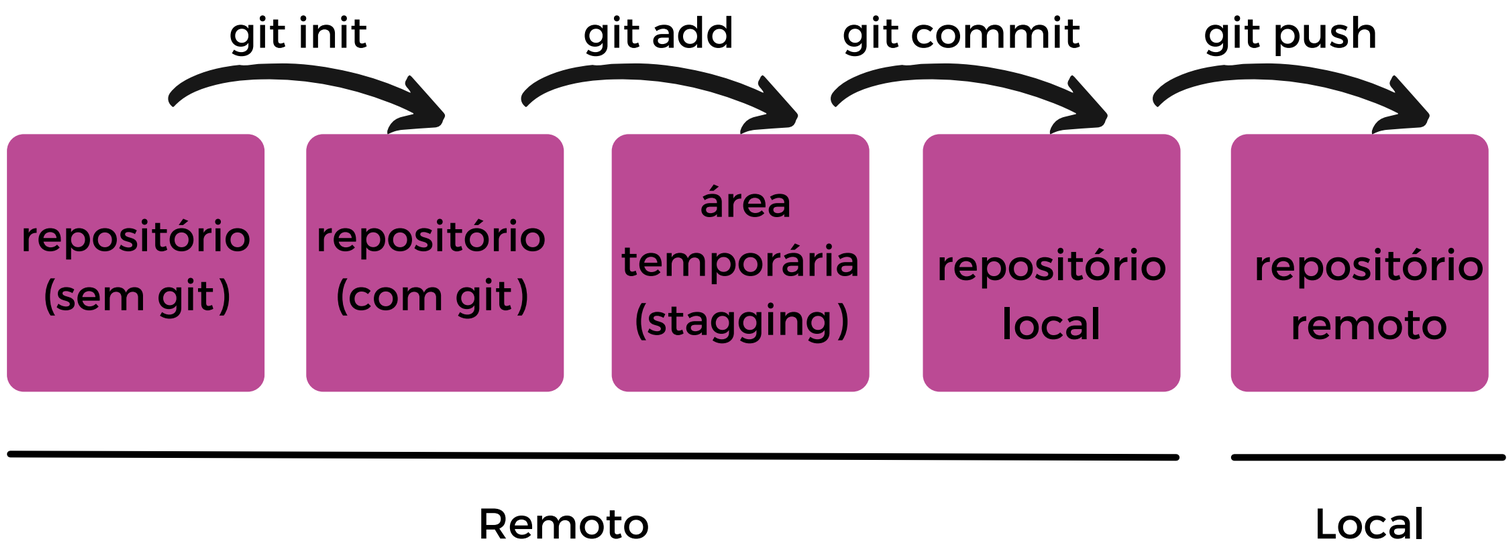


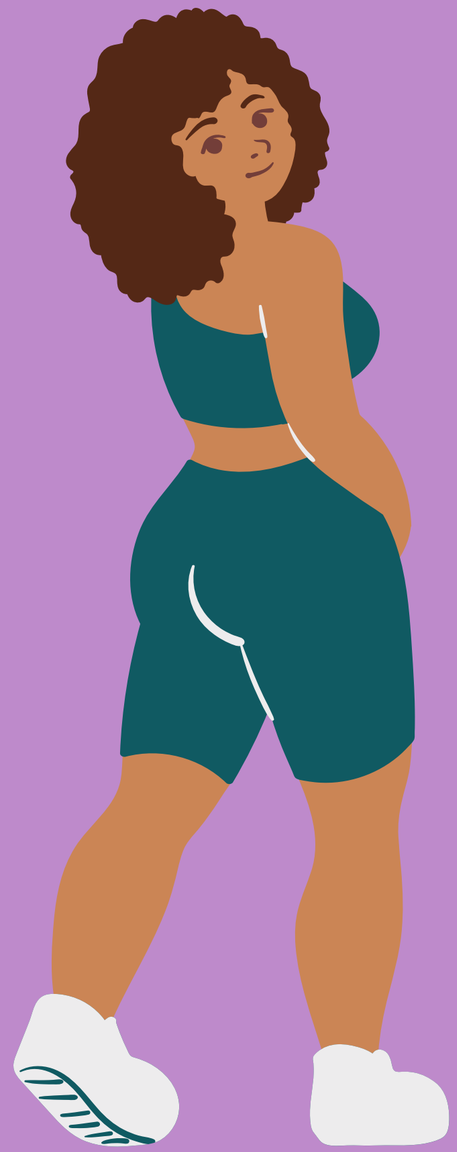
GitHub

GitHub é uma **plataforma de hospedagem** de código-fonte com controle de versão **usando o Git**.

Ele permite que **qualquer pessoa** cadastrada na plataforma contribua em projetos privados e/ou de código-fonte aberto (Open Source) de qualquer lugar do mundo.

Git, comandos básicos

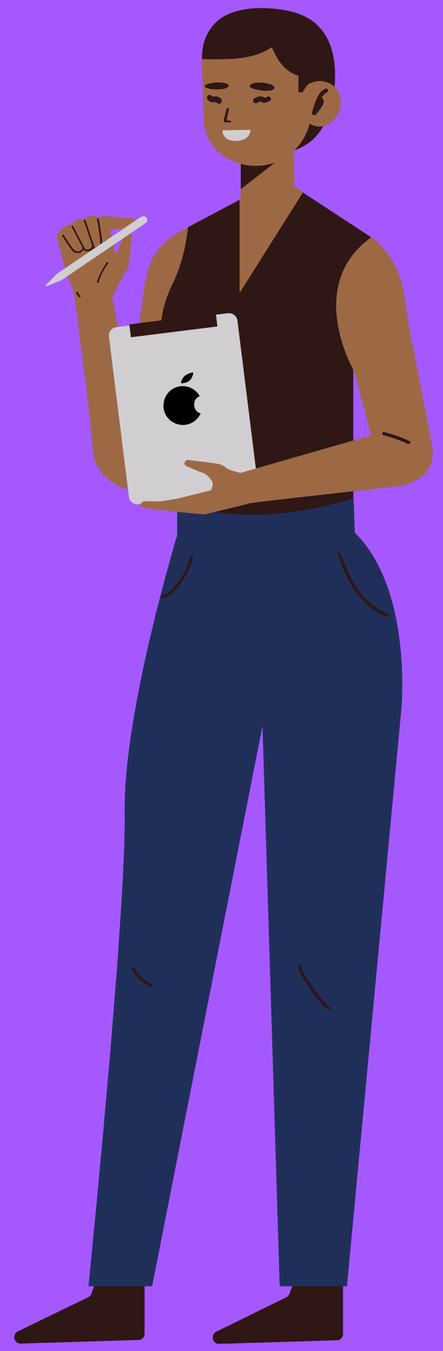




Comandos iniciais

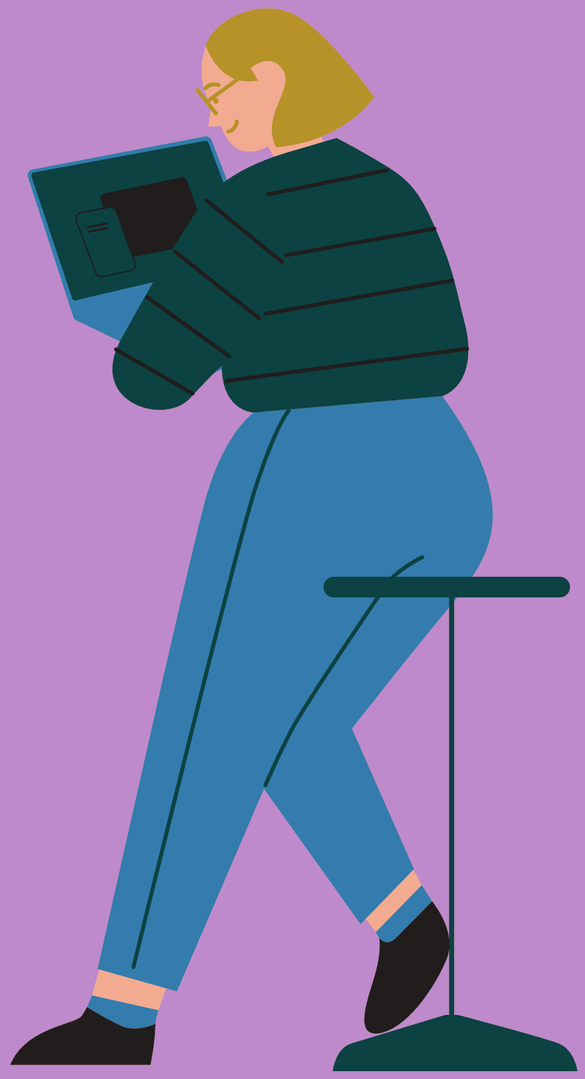
Git, conceitos básicos

repositório: pasta/local aonde o projeto é armazenado
clone: literalmente clonar(copiar uma versão) do repositório remoto o projeto para o nosso repositório local
branches:(galhos) uma parte muito útil no desenvolvimento coletivo, permite que cada usuário tenha seu “bracinho”(versão) dentro do projeto de maneira independente.
pull: puxar do repositório remoto para o repositório local as ultimas alterações.
commit: controla a versão de um arquivo, registrando através de uma mensagem que identifica as últimas alterações
push: puxar do repositório remoto para o repositório local as ultimas alterações



Git, conceitos básicos

merge: quando unificamos branches diferentes
fork: é uma cópia de um projeto para a sua conta do GitHub, é como se fosse uma xerox mesmo.
pull request: solicitação de merge da sua branch em um projeto de outra pessoa.
rebase: segue a linha de raciocínio do merge, mas apaga parte dos commits no histórico. Recomendado para ser usado entre branches de desenvolvovedores, não diretamente na branch principal por exemplo.



Exercícios

Exercício 1

Começando com Git, no terminal:

- Crie uma pasta
- Navegue até a pasta e inicialize o git
- Crie um arquivo qualquer e verifique seu status
- Adicione o arquivo ao stage do Git
- Faça um commit
- Faça um push

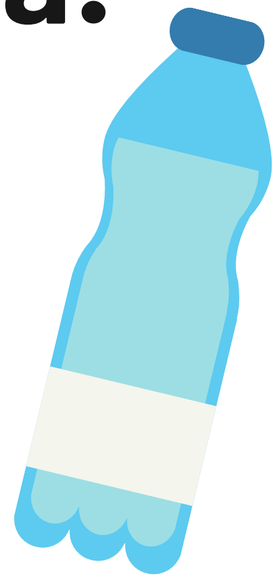
Exercício 2

Apresentação:

- Crie um repositório localmente e inicialize o git
- Adicione um arquivo markdown chamado README com seu nome e prato favorito e faça um commit
- Adicione uma curiosidade sobre você e faça outro commit
- Publique o repositório no seu GitHub

Vamos fazer uma pausa rápida!

Voltamos em 10 minutos, e não se esqueçam de beber àgua



Aprofundando no Git

Configurações iniciais

```
git config --global user.name "mandypry"
git config --global user.email “amanda.adgti@gmail.com”
git config --list
git config --global --unset user.name "Nome"
git config --global --unset user.email “nome@email.com”
```

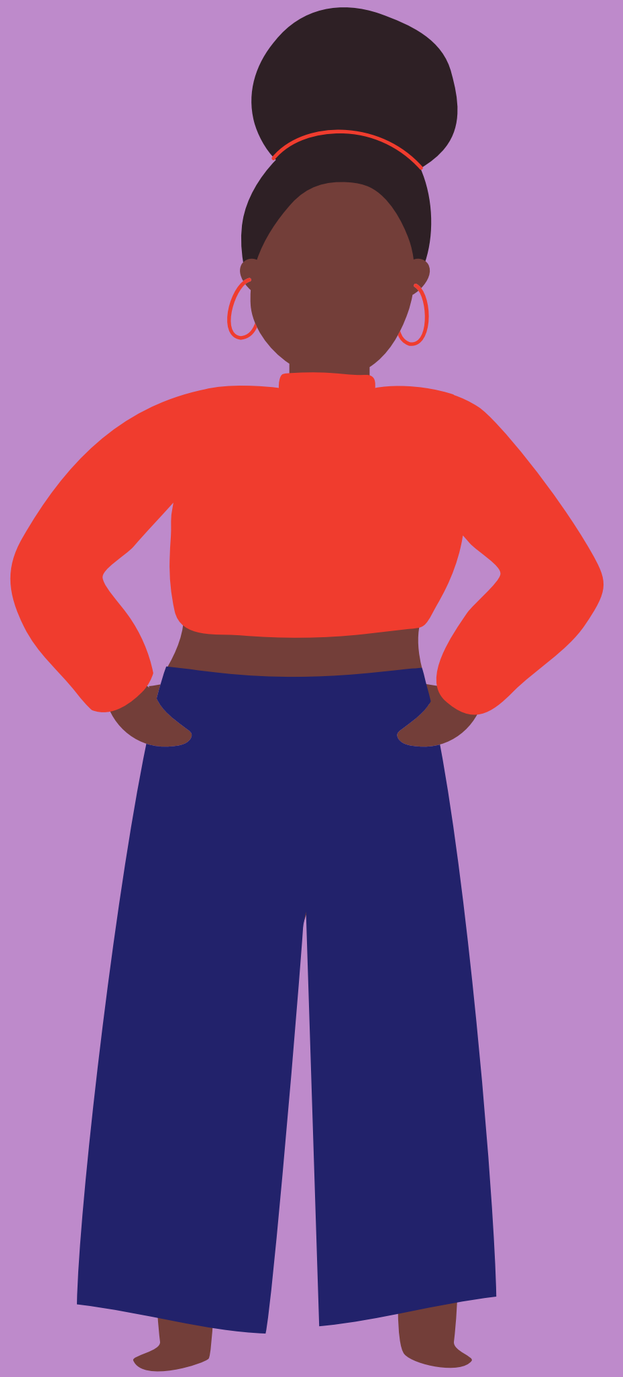
Exercício

Algoritmos:

- Faça um fork do repositório.
- Clone o repositório para a sua máquina.
- Crie uma nova branch com seu nome (exemplo: amanda-silva).
- Faça commits com a resolução dos exercícios.
- Atualize seu repositório remoto.

DESAFIO EXTRA:

- Abra um Pull Request para o repositório original

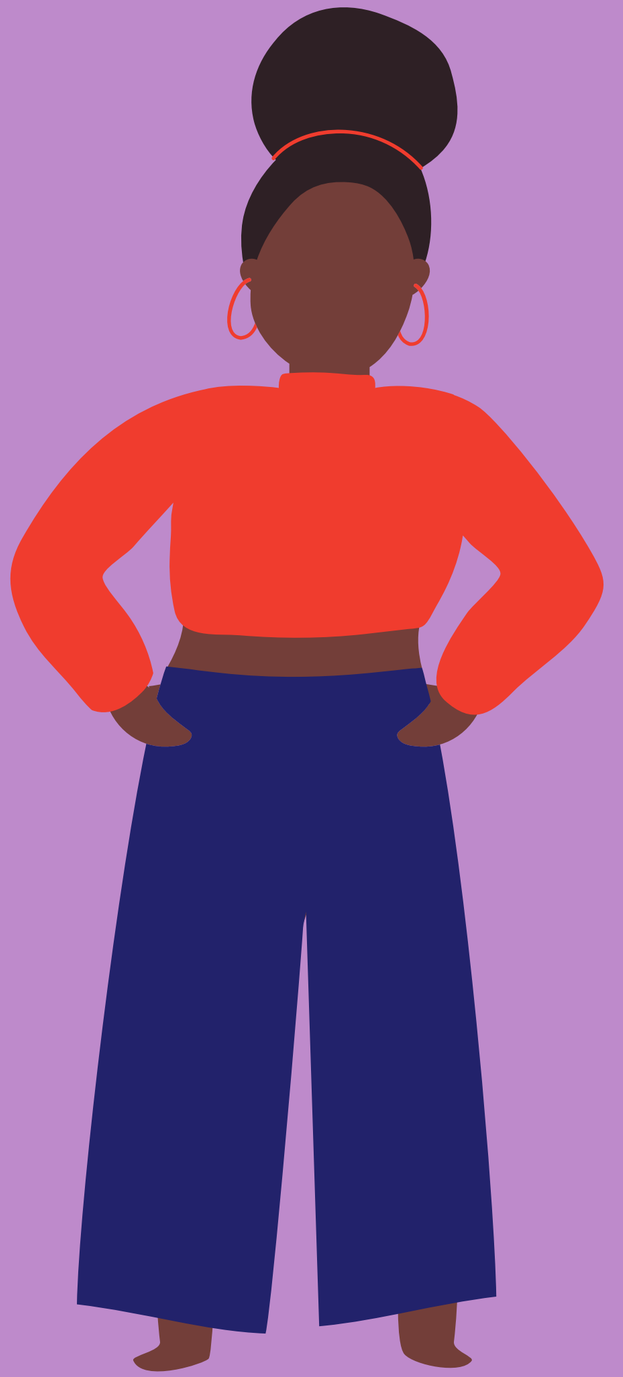


Git, conceitos básicos

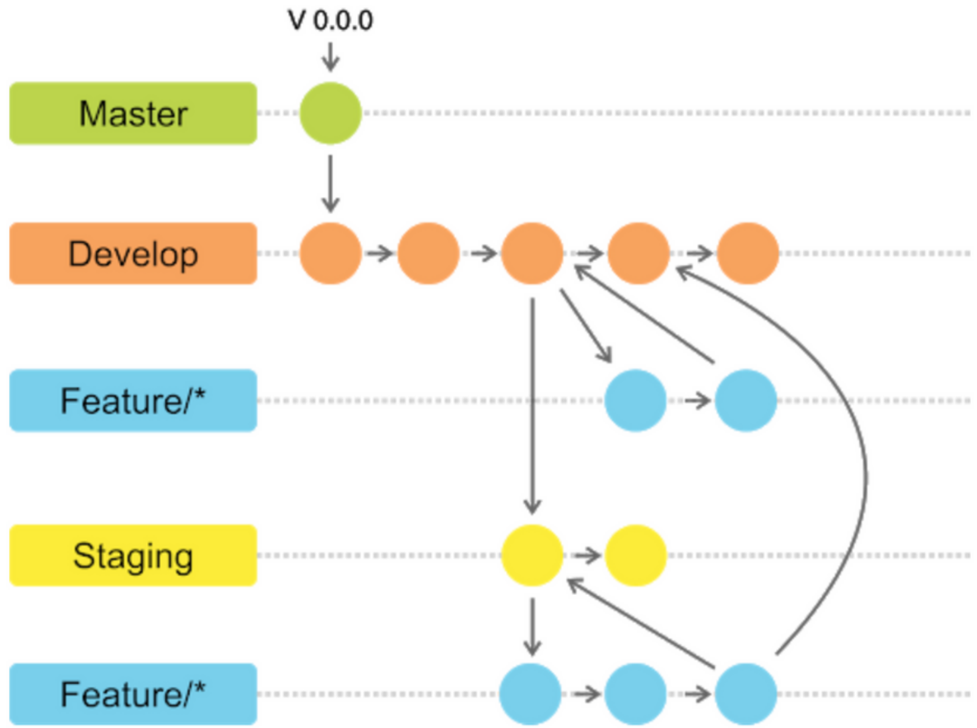
git init
git add ou git add nome-do-arquivo
git commit -m "mensagem do commit"
git status
git remote add origin url-do-repo ou ssh-do-repo
git remote -v
git push origin nome-da-branch
git clone url-do-repo

Git, sobre branches

```
git branch
git checkout -b nome-da-branch
git checkout nome-da-branch
git branch -d nome-da-branch
git push origin --delete nome-da-branch
git merge nome-da-branch
```



Git, sobre branches



Git, sobre forks

git clone url-do-seu-fork
git remote add upstream url-do-repo-original
git fetch (ou pull) upstream
git rebase (ou merge) upstream/main

Exercícios

Exercício 1

Faça um fork do repositório -> github.com/mandyppry/reprograma-15
A partir do seu fork, seu link, faça um clone
Veja se o repositório remoto é o seu fork
Cria uma branch com seu nome
Crie um arquivo com seu nome e escreva algo dentro
Adicione esse arquivo ao olhar do git
Faça um commit
Adicione o repositório que forkamos como atualizador
Atualize o seu repositório remoto com o local

Exercício 1, continuação

No seu repositório/pasta local:

- Crie uma branch com seu nome
- Crie um arquivo com seu nome e escreva algo dentro
- Adicione esse arquivo ao olhar do git
- Faça um commit
- Faça um git push para o seu repositório
- No seu repositório no GitHub faça um pull request entre forks, ou seja envie a sua branch para o do repositório original

Exercício 2

- Crie um repositório no GitHub, sem o Readme.md
- Crie uma pasta no seu computador
- Abra o git bash nela
- Adicione o seu repositório remoto como o repositório de origem
- Dê o comando git init
- Crie uma branch com seu nome
- Dentro da pasta crie um arquivo e escreva algo
- Adicione o arquivo ao git
- Faça um commit
- Envie as alterações para o seu repositório remoto

No último passo colocar o nome da sua branch local após o origin !!

Exercício 1, continuação

Entre no repositório github.com/mandyppry/reprograma-15
- Vá na aba Pull Requests
- Procure a sua PR, clique nela
- Clique em merge pull request

Exercício para casa

- Faça um clone no seu pc -> https://github.com/reprograma/ON15-TET-S1-GIT
 - Entrar na pasta do clone
 - Cria uma branch com seu nome
 - Crie um arquivo com seu nome e escreva uma música, um lugar e um livro ou disco favorito seu
 - Adicione esse arquivo ao olhar do git
 - Faça um commit
 - Faça um git push para repositório
 - No seu repositório no GitHub faça um Pull Request, ou seja envie a sua branch para a branch do repositório original

Obrigada pela presença maravilhosas!

amanda-silva-dev

amanda.adgti@gmail.com