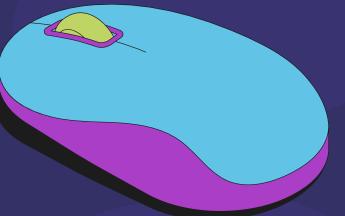
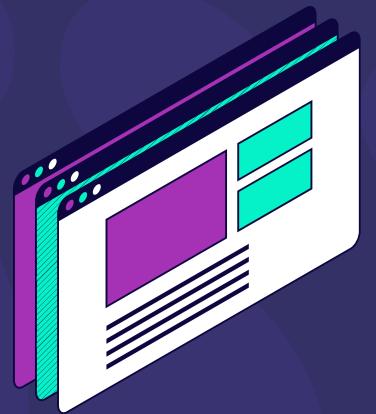




Vamos  
reprogramar  
o mundo?



*Bem-vindas ao*  
**Todas em Tech(TeT)**  
**turma on19**





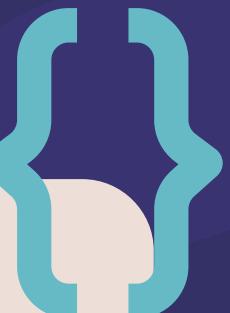
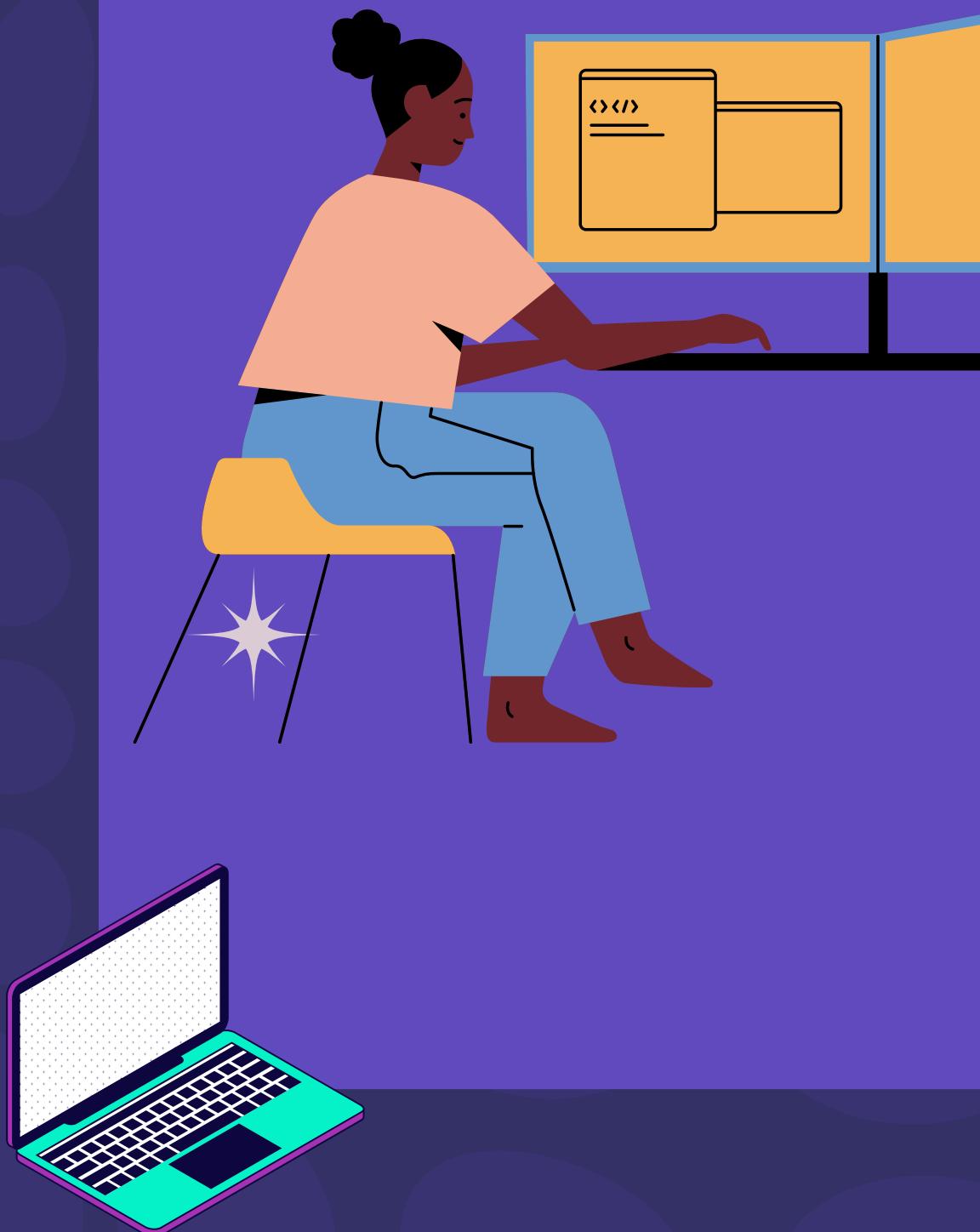
**Louise Costa  
(Loui)**



Professora, Eletricista,  
Desenvolvedora  
Backend, Carioca,  
Gamer e Otaku



# Apresentação das monitoras





# {reprograma}

## Introdução a programação, Git & GitHub - On19



# Regrinhas:



**Use a reação do Zoom para levantar a mão para qualquer dúvida!**

**Tem dúvida? Pergunte!**

**Tá com vergonha de perguntar? Pergunte assim mesmo!**

**Não se incomode em perguntar as monitoras pelo chat.**

**Se puder deixe a câmera ligada!**

**Deixe apenas o necessário aberto no seu computador!**

**Beba água e divirta-se. Não se apresse pequena padawan!**

# Vamos começar?



## Agenda:

- Conceitos básicos - o que é programação?
- Backend, Frontend, Fullstack e Mobile
- Algoritmo & Pseudocódigo
- Terminal
- Versionamento de código
- Git + GitHub
- Comandos Básicos do Git
- Meu primeiro repositório
- Tarefa de casa

# Conceitos básicos

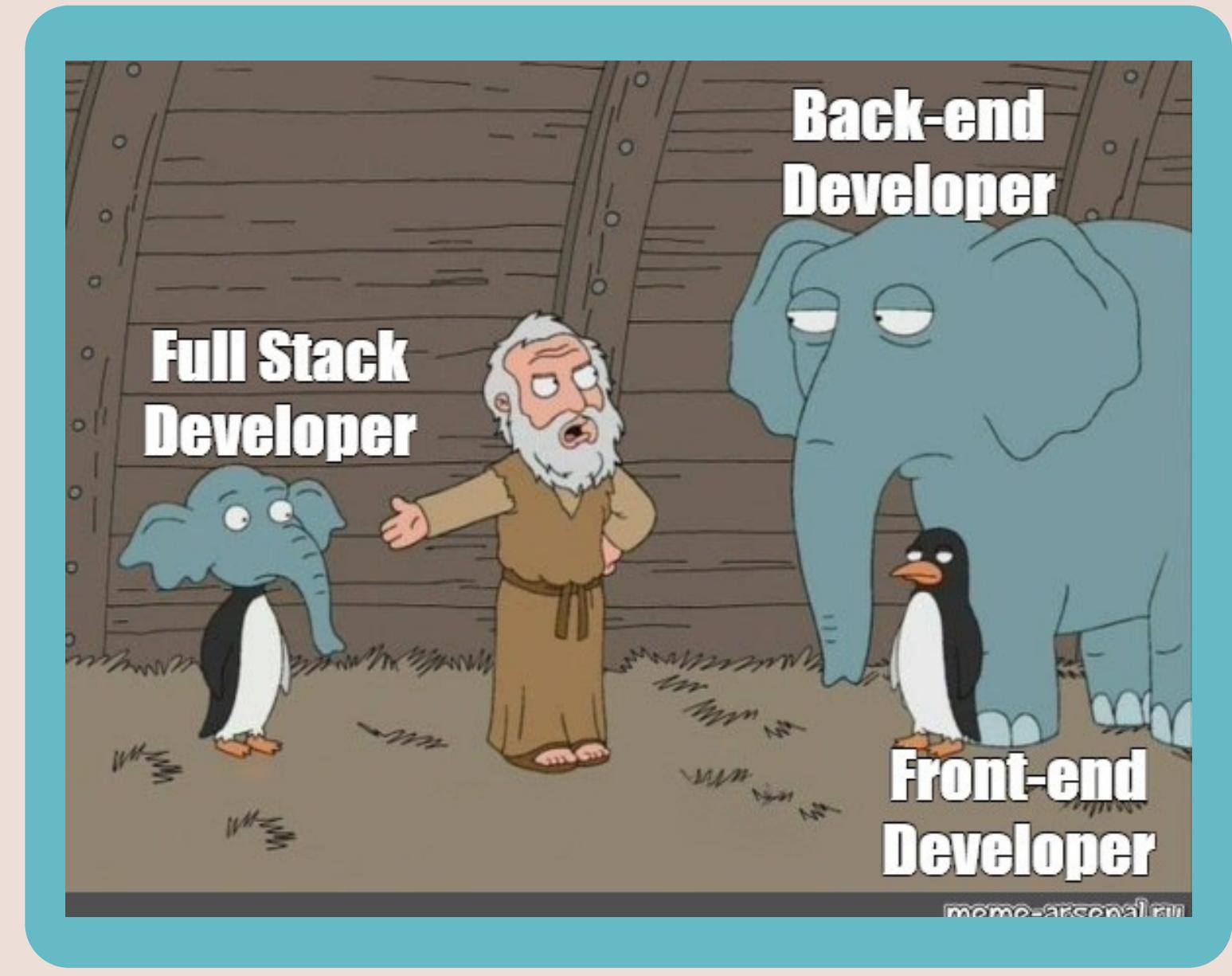
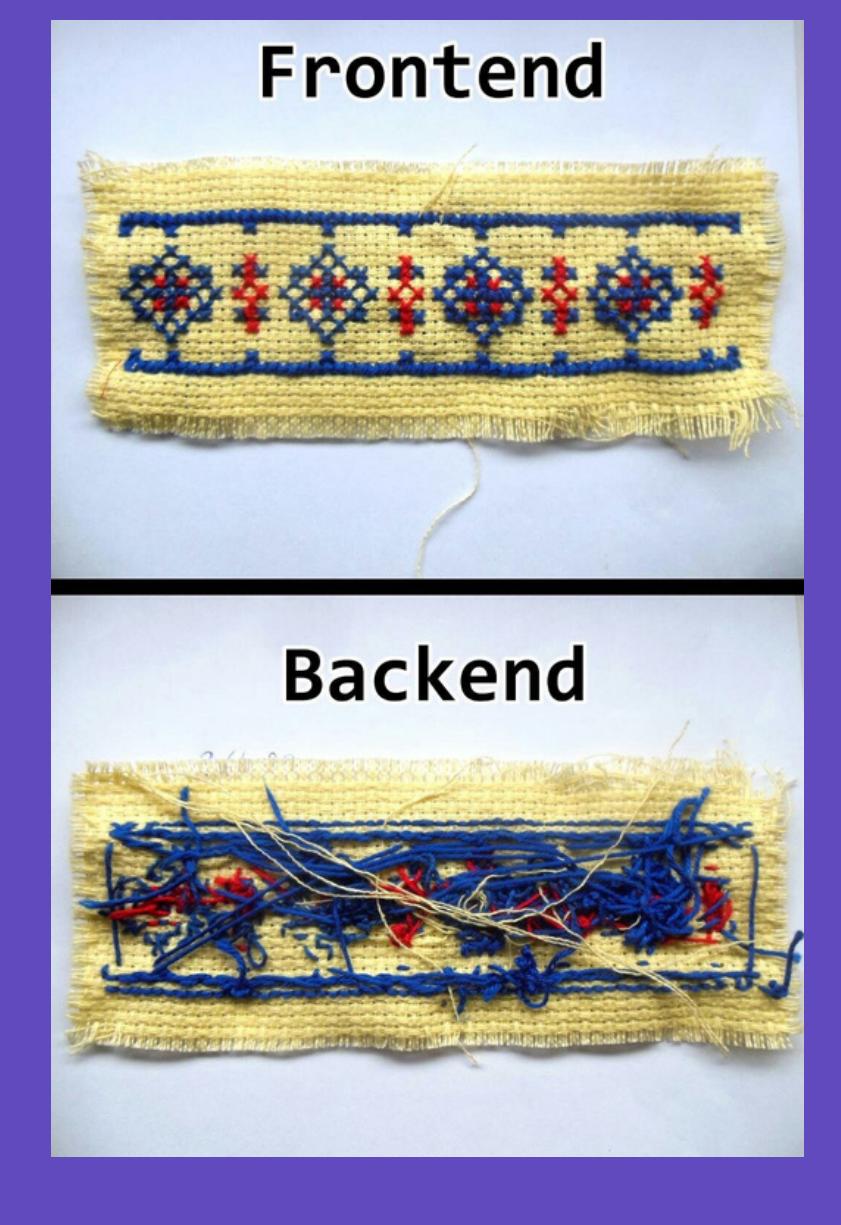
## O que é programação?



- É um processo de escrita, testes e manutenção de programas de computadores.
- Ada Lovelace foi a primeira programadora!



# *Backend, Frontend, Fullstack, Mobile...*



# *Frontend*

- É responsável pelo visual da aplicação, a parte onde nós interagimos e vemos.

# *Backend*

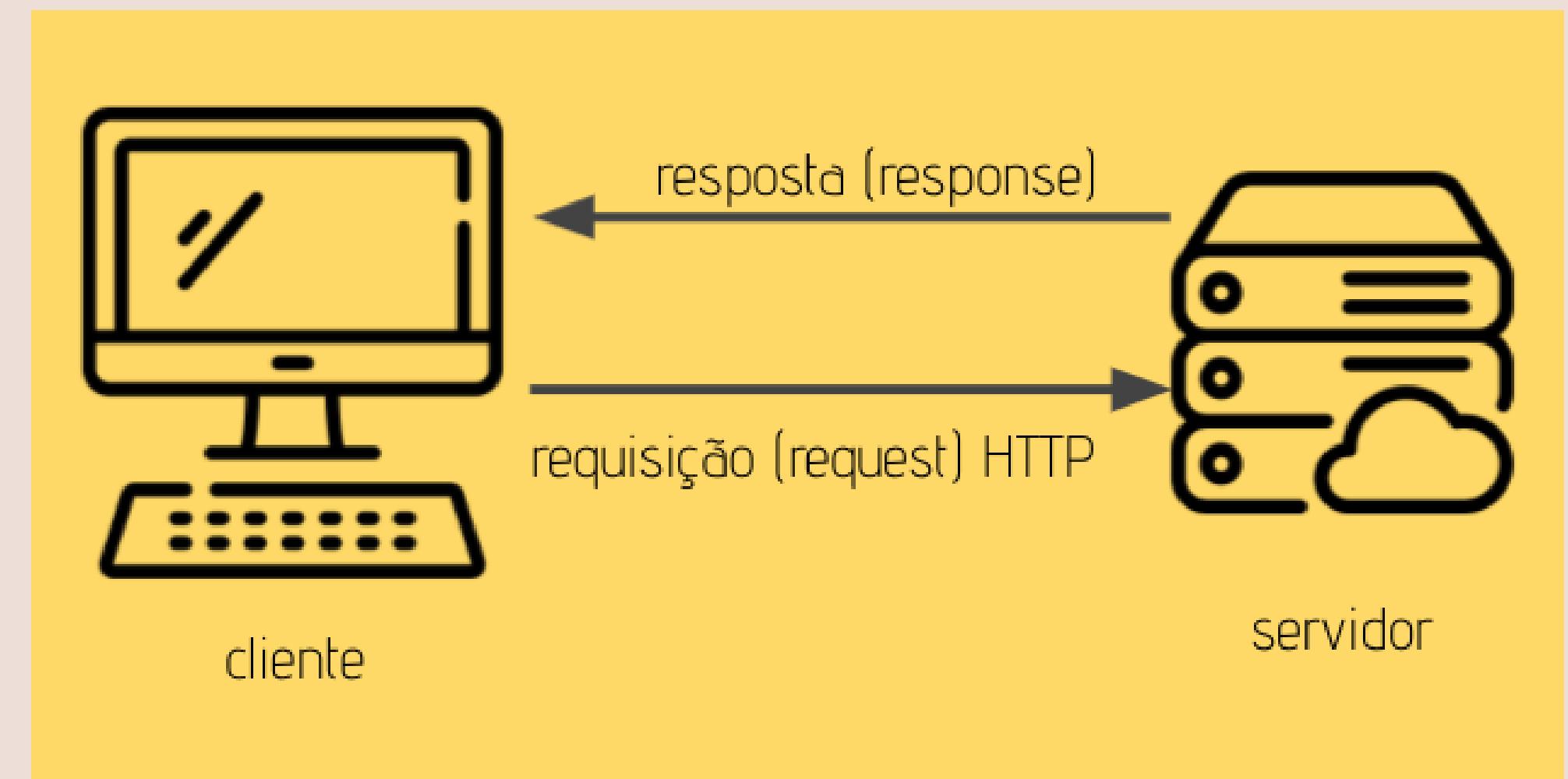
- É a parte por trás de uma aplicação. É responsável pela operação e não interage com a usuária diretamente. Geralmente faz a ponte entre o frontend e o banco de dados

# *Banco de dados*

- É um agrupamento de dados que são relacionados ao mesmo local e que precisam ser armazenados para segurança e acesso em qualquer hora.

# *Client-side & Server-side (ou cliente -servidor)*

- Uma aplicação web é composta de Client (cliente) e Server(servidor).
- O client nada mais é do que o nosso browser, ou navegador, ele que inicia a comunicação com o servidor.
- O server é o servidor, o papel dele é receber uma requisição e devolver uma resposta pro cliente.





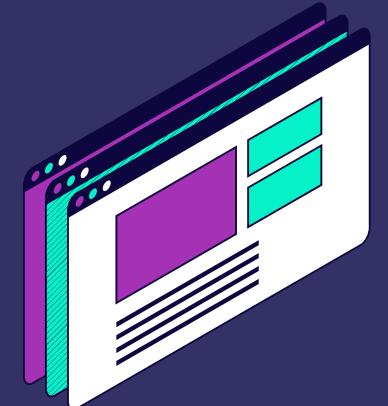
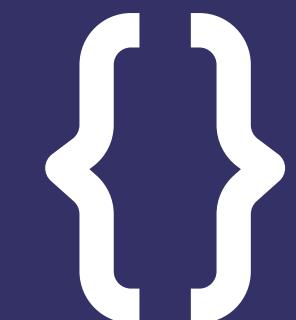
# Algoritmo & Pseudocódigo

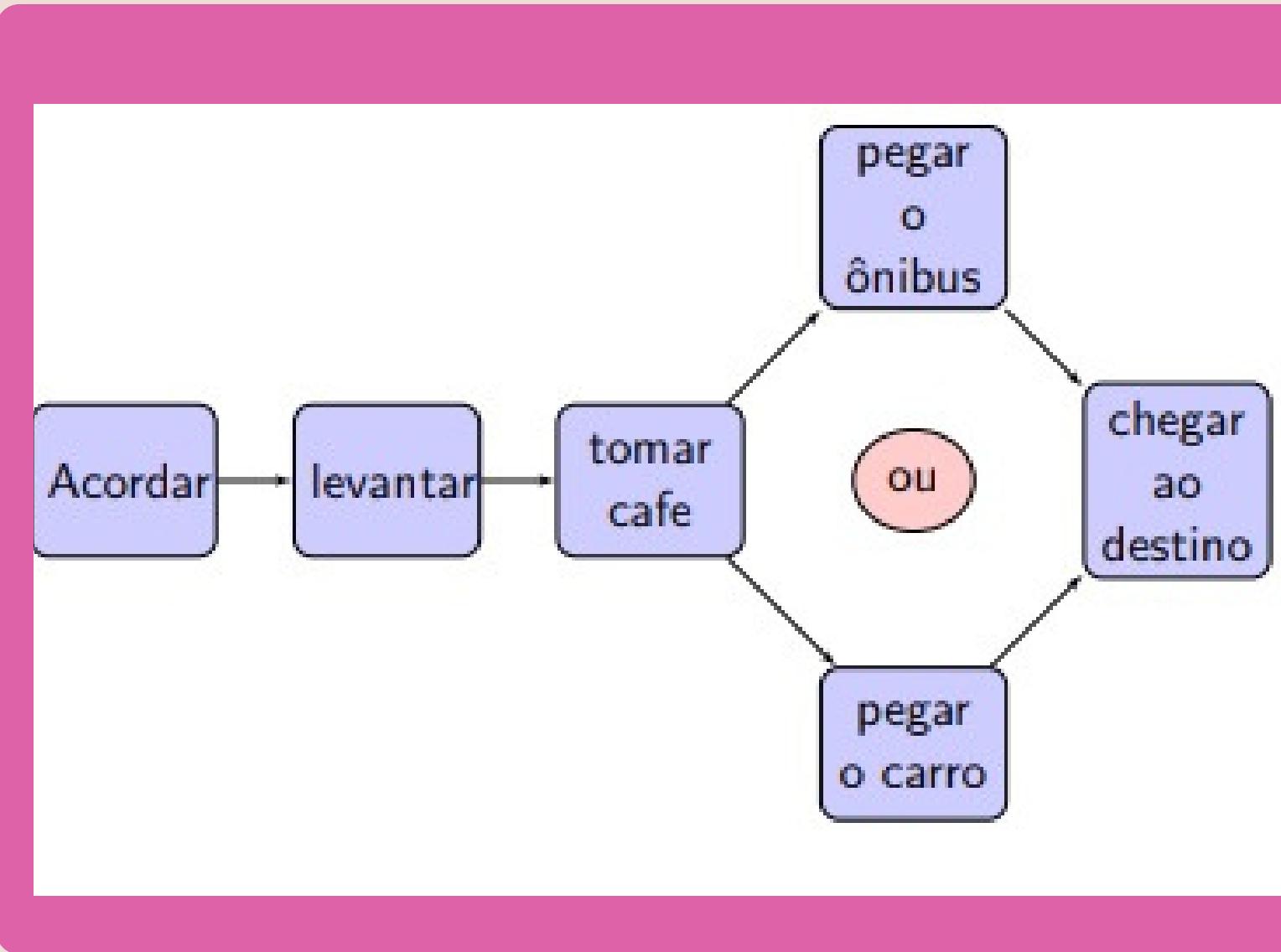
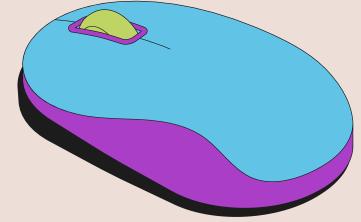
## Algoritmo:

- sequência finita de cálculos que leva a resolução de um problema
- Ele pode ser chamado de passo a passo também
- O algoritmo possui entrada, processamento e saída

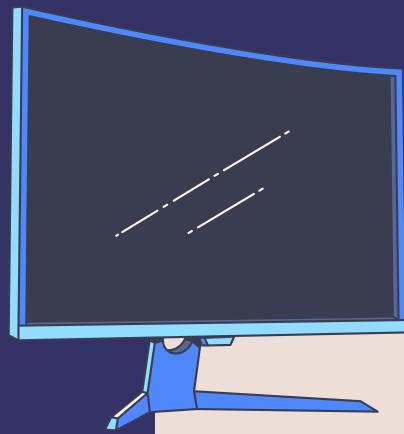
## Pseudocódigo:

- É a representação de um algoritmo em um texto que explique um passo a passo, em uma lista com tarefas ordenadas ou até mesmo em um fluxograma.





```
1. início
2. // declaração de variáveis
3. real: N1, N2, N3, N4, // notas bimestrais
4. MA; // média anual
5. leia (N1, N2, N3, N4); // entrada de dados
6. MA ← (N1 + N2 + N3 + N4) / 4; // processamento
7. escreva (MA); // saída de dados
8. se (MA >= 7)
9. então
10.     escreva ("Aluno Aprovado!");
11. fimse;
12. fim.
```

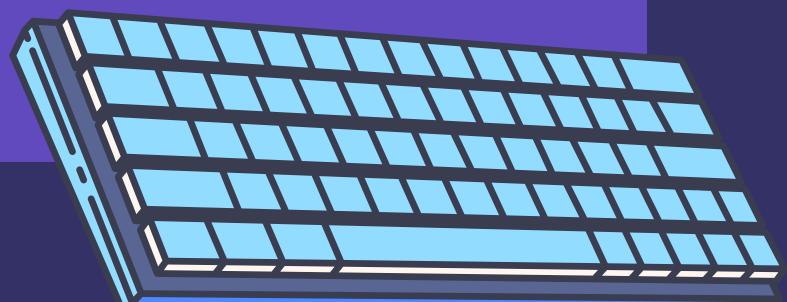


## Trocar uma lâmpada

1. Pegar uma escada
2. Pegar uma lâmpada nova
3. Subir na escada
4. Tirar a lâmpada antiga
5. Rosquear a lâmpada nova
6. Descer da escada
7. Ligar o interruptor

## Fazer café

1. Pegar a chaleira
2. Colocar a água
3. Pegar o coador
4. Pegar o pó de café
5. Colocar o pó no coador
6. Adiciona a água quente
7. Pegar a caneca
8. Toma café! Somos pessoas felizes

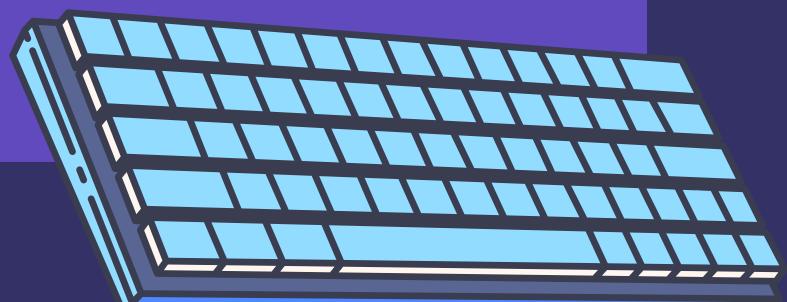




# Terminal

- Programa que usamos para gerenciar recursos mais avançados do sistema
- Podem ser conhecidos como CLIs (Command-line interface, ou Interface de Linha de Comando).

```
MINGW64:/c/Users/Felipe/Documents/MyProjects/GitProjects/simple-memory-helper
Felipe@DESKTOP-NT07TTQ MINGW64 ~
$ cd Documents
Felipe@DESKTOP-NT07TTQ MINGW64 ~/Documents
$ cd gitprojects
bash: cd: gitprojects: No such file or directory
Felipe@DESKTOP-NT07TTQ MINGW64 ~/Documents
$ cd My
My Music/ My Pictures/ My Videos/ MyProjects/
Felipe@DESKTOP-NT07TTQ MINGW64 ~/Documents
$ cd My
My Music/ My Pictures/ My Videos/ MyProjects/
Felipe@DESKTOP-NT07TTQ MINGW64 ~/Documents
$ cd My
My Music/ My Pictures/ My Videos/ MyProjects/
Felipe@DESKTOP-NT07TTQ MINGW64 ~/Documents
$ cd My
My Music/ My Pictures/ My Videos/ MyProjects/
Felipe@DESKTOP-NT07TTQ MINGW64 ~/Documents/MyProjects
$ cd GitProjects/
Felipe@DESKTOP-NT07TTQ MINGW64 ~/Documents/MyProjects/GitProjects
$ cd simple-memory-helper/
Felipe@DESKTOP-NT07TTQ MINGW64 ~/Documents/MyProjects/GitProjects/simple-memory-helper (master)
$ git pull
Already up-to-date.
```



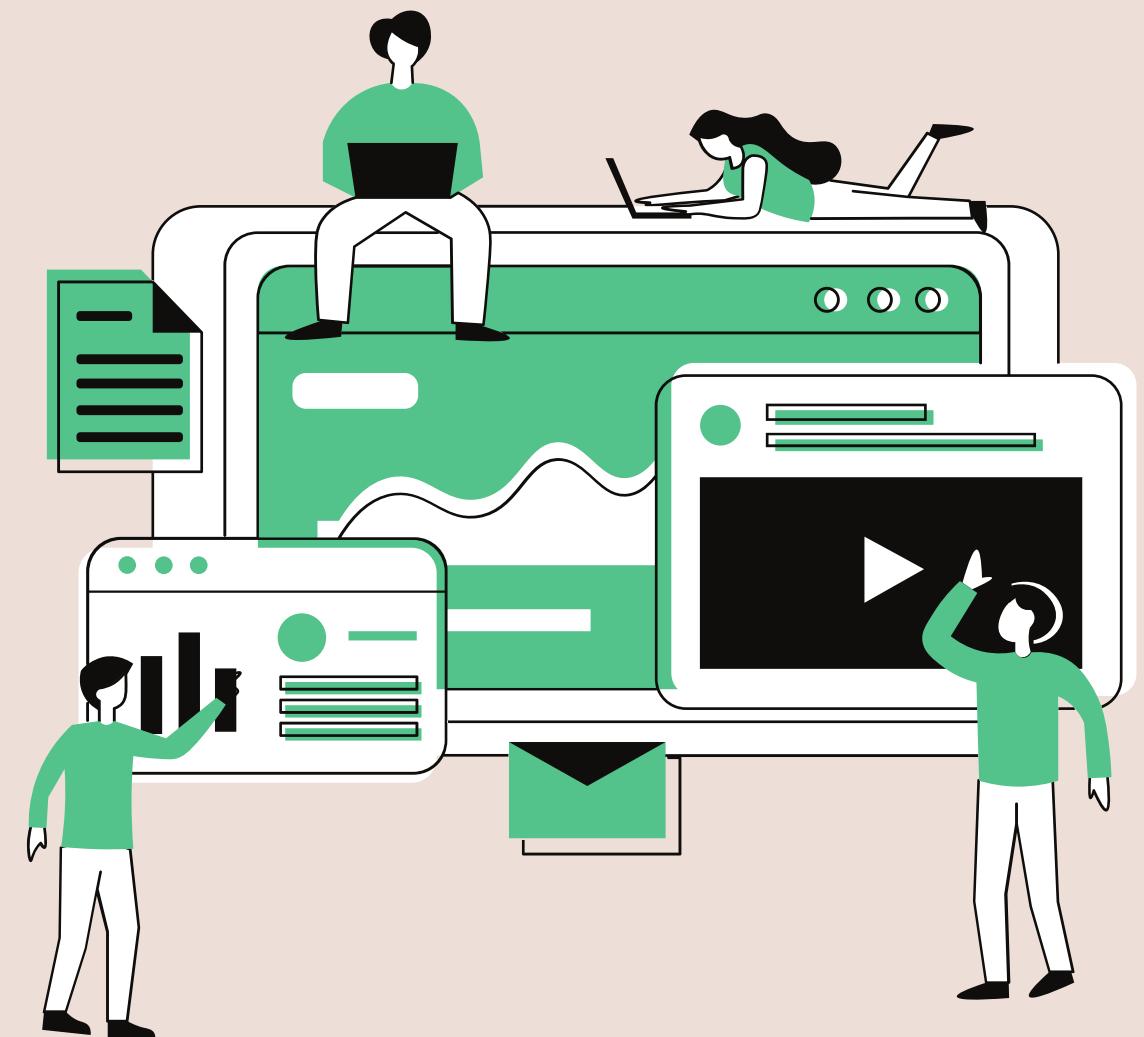
# Versionamento de código

Nome	Data de modificação	Tipo	Tamanho
Desenhos	20/07/2022 00:31	Pasta de arquivos	
Filmes	22/07/2022 16:16	Pasta de arquivos	
Imagens README	10/08/2022 17:24	Pasta de arquivos	
LGBT	19/07/2022 22:54	Pasta de arquivos	
Livros	20/07/2022 00:25	Pasta de arquivos	
Seriados	20/07/2022 00:46	Pasta de arquivos	
projeto-final	10/08/2022 17:25	Documento do Mi...	13 KB
projeto-final-repo	10/08/2022 17:25	Documento do Mi...	13 KB
projeto-final-repo1	10/08/2022 17:25	Documento do Mi...	13 KB
projeto-final-repo2	10/08/2022 17:25	Documento do Mi...	13 KB
projeto-final-repo3-final	10/08/2022 17:25	Documento do Mi...	13 KB
projeto-final-repo3-final-certo	10/08/2022 17:25	Documento do Mi...	13 KB

**Quem nunca fez  
isso que atire a  
primeira pedra!**



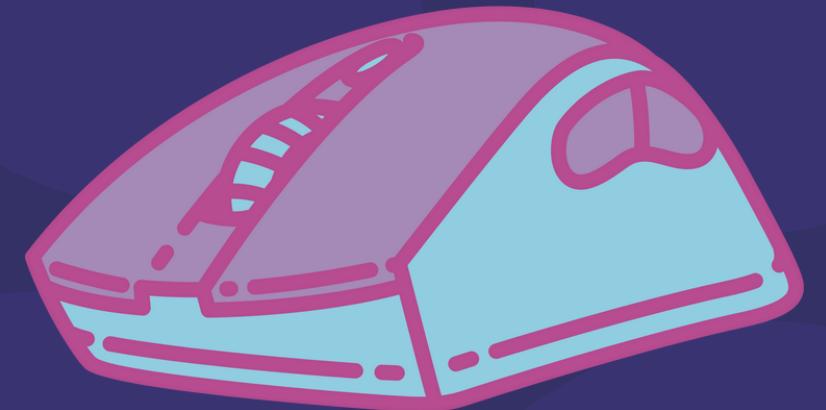
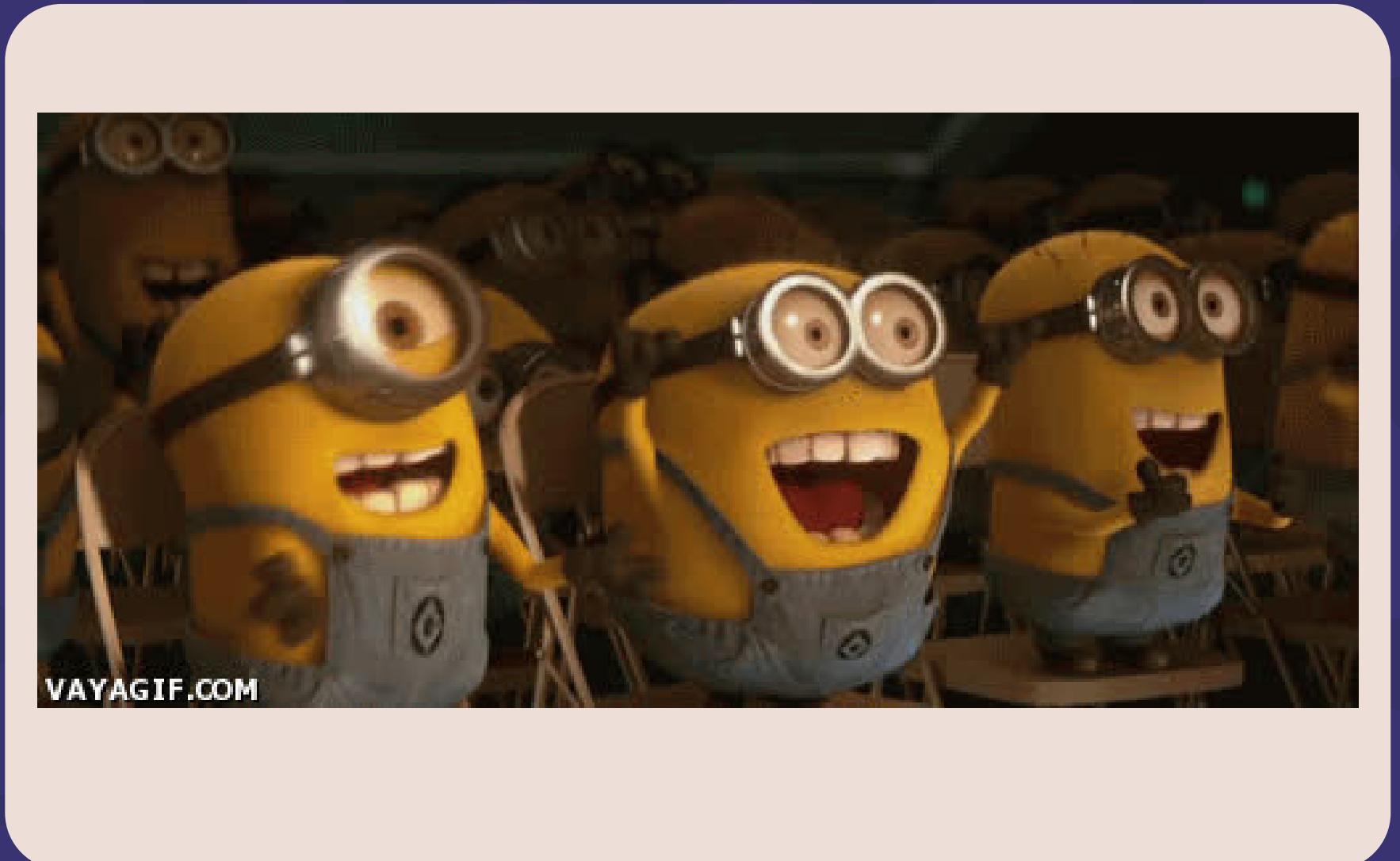
# *Controle de versão*



- Histórico de alterações completo e a longo prazo de todos os arquivos.
- Ramificação e mescla.
- Rastreabilidade.



E agora...



# Git



Quem ou o que é o Git???

1. Sistema de controle de versões, usado principalmente no desenvolvimento de software;
2. Pode-se desenvolver projetos colaborativos com diversas pessoas trabalhando simultaneamente
3. Guarda o histórico de tudo que foi alterado no arquivo ao longo do tempo, além de mostrar quem fez a mudança.

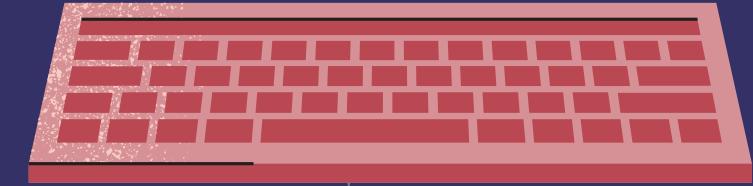


# Instalando o Git

The image shows three screenshots illustrating the Git installation process. On the left, the official Git website's 'Downloads' page is displayed, featuring links for macOS, Windows, and Linux/Unix. In the center, the 'Git 2.37.1 Setup' window titled 'Select Components' shows a list of checkboxes for various Git features, with the first two (Additional icons and On the Desktop) unselected. On the right, another window titled 'Adjusting the name of the initial branch in new repositories' shows a radio button selected for 'Override the default branch name for new repositories', with the input field containing the value 'main'.

- No site oficial do Git, baixe a versão para o seu SO <https://git-scm.com/downloads>;
- Execute o arquivo baixado e vá dando “Next” até a tela “Select Components”. Só deixe as duas primeiras opções desmarcadas;
- Na tela de branch, escolha a 2ª opção e escreva **main**
- Agora aperte next até o final

# Comandos básicos do Git



**git config user.name "Nome da Reprogramer":** configura o seu nome

**git config user.email "emailDaReprogramer@gmail.com":** configura o seu email

**git config --list:** mostra todas as propriedades de configuração do Git

**ls:** listar (ele traz uma lista de tudo o que está naquela pasta - documentos, outras pastas, etc)

**mkdir nome-da-pasta:** cria uma pasta

**touch nome-do-arquivo:** cria um arquivo

**cd:** usado para se locomover entre as pastas

**cd ~ :** volta para a pasta raiz

**cd .. :** volta uma pasta

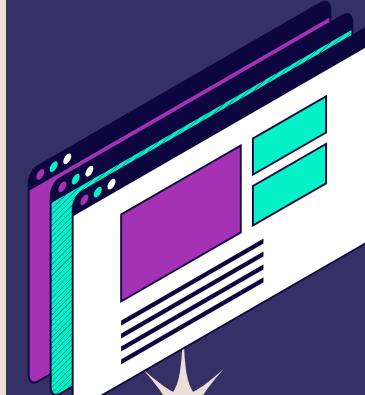
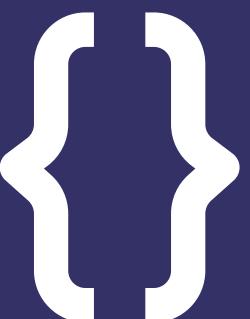
**pwd:** traz o caminho onde você está (em que pasta e onde essa pasta fica)

**rm arquivo:** remove, deleta um arquivo.

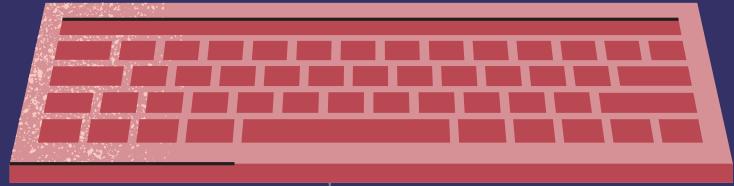
**rm -f ou rm --recursive pasta:** deleta uma pasta

**whoami:** "quem sou eu?" identifica o usuário que está mexendo no sistema

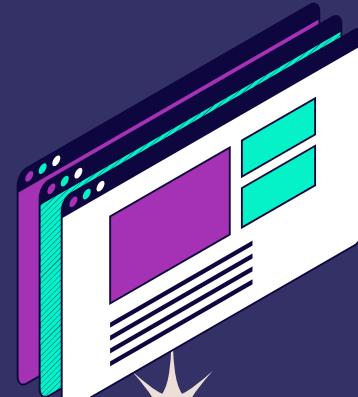
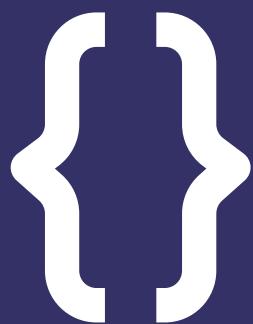
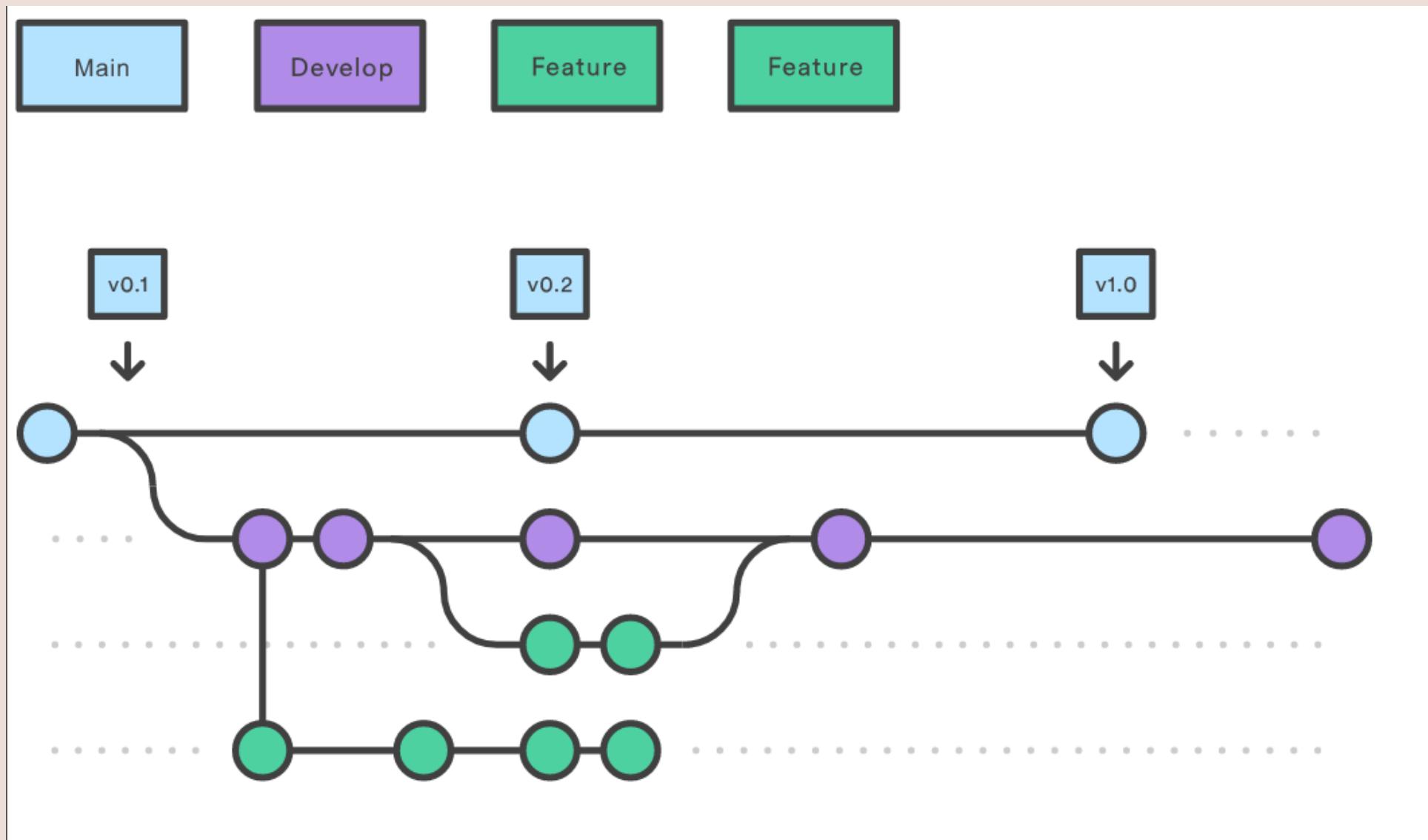
**git init:** inicializa o git no repositório local

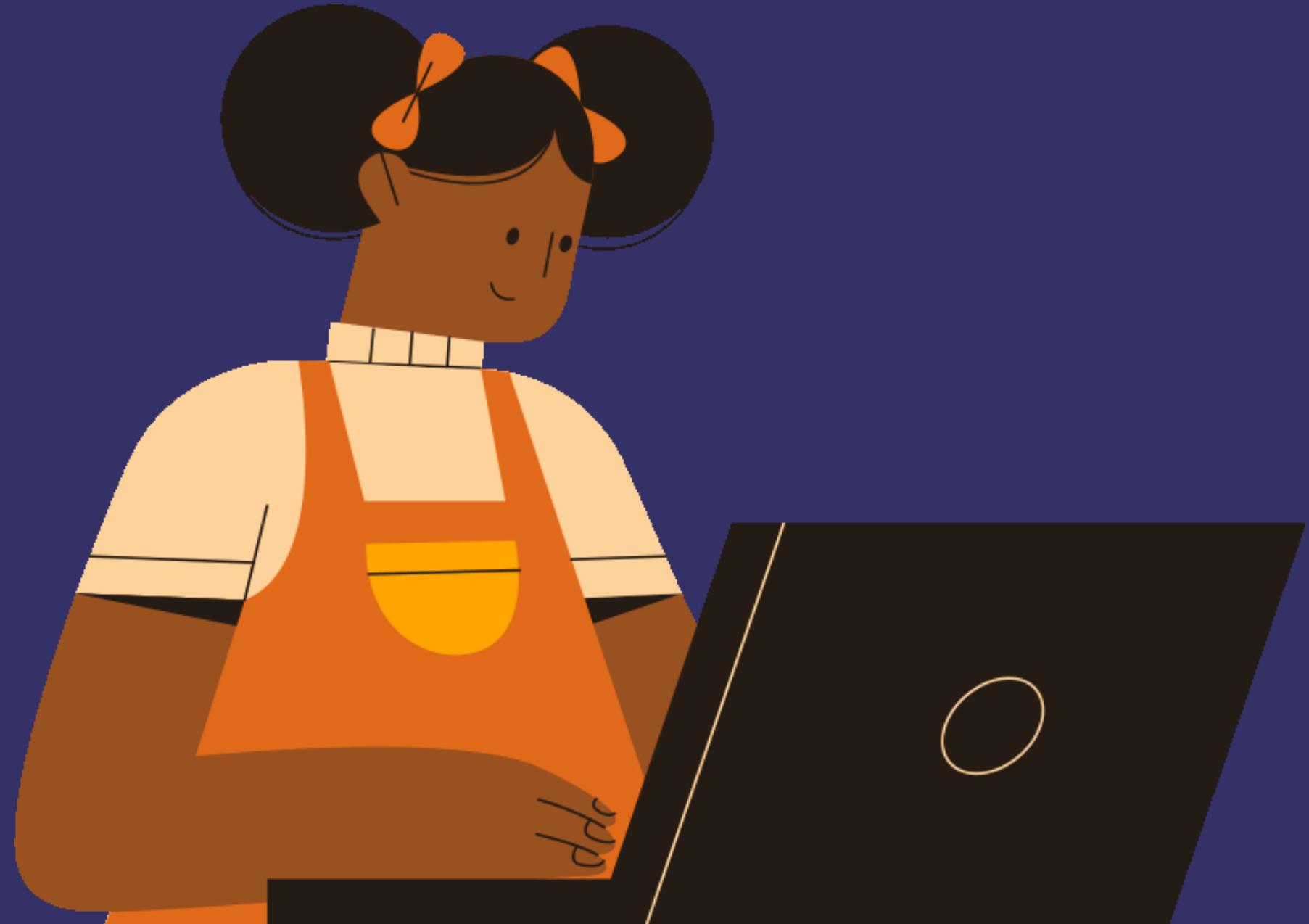


# O que é Gitflow?



É um modelo alternativo de ramificação do Git que consiste no uso de ramificações de recursos e várias ramificações primárias.





## Vamos colocar a mão na massa?

- Abra o Git Bash
- Identifique o usuário
- Confirme a pasta que você está
- Volte uma pasta
- Crie uma pasta chamada **tempo**
- Entre na pasta
- Crie um arquivo chamado **temporario**

# GitHub



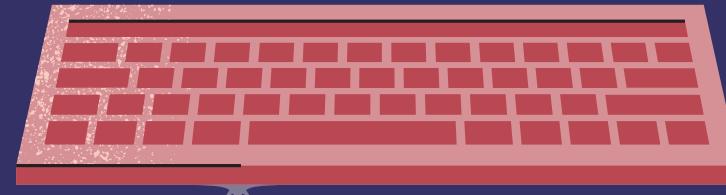
- Plataforma de hospedagem de código-fonte com controle de versão usando o Git.
- É amplamente utilizado por programadores para divulgação de seus trabalhos ou para que outros programadores contribuam com o projeto.



Agora vamos  
criar a nossa  
conta no  
GitHub



# Comandos básicos



**git add nome-do-arquivo:** adiciona um arquivo modificado ao staging (área temporária)

**git add . :** adiciona todos os arquivos modificados ao staging (área temporária)

**git status:** mostra os status dos arquivos modificados

**git commit -m "mensagem":** cria um commit

**git pull:** puxa as atualizações mais recente (remoto -> local)

**git push:** envia as atualizações mais recentes (local -> remoto)

**git remote add origin nome-da-branch:** faz um link entre seu repositório local com o remoto

**git checkout -- nome-arquivo:** descarta as alterações locais do arquivo informado

**git checkout -b nome-da-branch:** cria uma branch a partir da atual

**git checkout nome-da-branch:** troca de branch

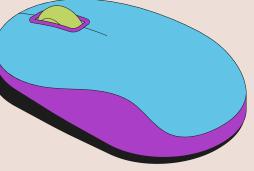
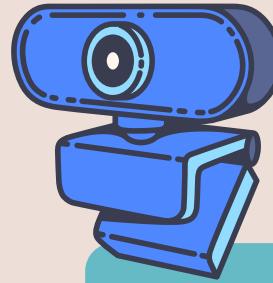
**git merge nome-da-branch:** mescla a branch passada no parâmetro com a atual

**git remote -v:** mostra as URLs para onde o git está apontando

**git log:** mostra o histórico de commits, com data, hora, mensagem e autora (caso fique presa nessa lista, aperte "q" para sair)

**git branch:** lista todas as branches locais

**git diff:** mostra no terminal a diferença entre os arquivos editados localmente



# Conceitos básicos

**Repositório:** espaço digital aonde o seu projeto vai ser salvo. No seu computador ele é a pasta aonde o seu projeto está salvo.

**Commit:** controle de versão (histórico) daquele arquivo e cria uma etiqueta para facilitar o entendimento do que foi salvo naquele momento.

**Pull:** serve para se comunicar entre a sua máquina e o repositório remoto. Esse comando faz uma cópia do repositório remoto e baixa ele para a sua máquina.

**Push:** serve para se comunicar entre a sua máquina e o repositório remoto. Esse comando faz uma cópia do repositório local e envia ele para o repositório remoto.

**Clone:** faz exatamente o que ele sugere: uma cópia exata do arquivo, que você vai baixar do repositório remoto para a sua máquina.

**Branch:** permite que cada usuário tenha o seu "galho" dentro de um projeto de maneira independente. Ela é uma parte muito útil no desenvolvimento em um projeto coletivo.

**Merge:** unifica diferentes branches

**Fork:** cria uma cópia de um projeto para o seu GitHub

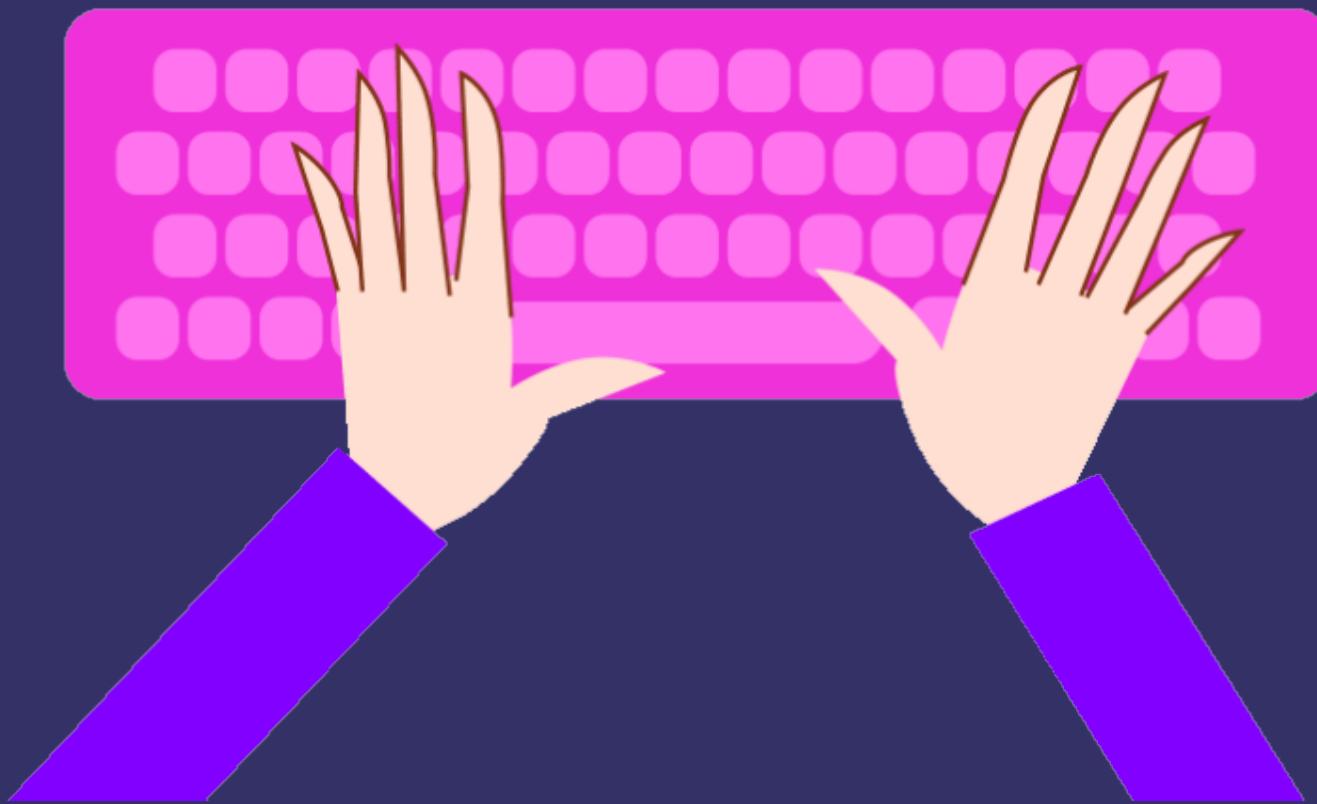
**Pull Request(PR):** solicitação de dar merge da sua branch em um projeto de outra pessoa

# Commits semânticos



- Commit semântico ou conventional commit, é uma das formas que se pode fazer padronização de commits dentro de um projeto de desenvolvimento de software.
- Utilizando regras simples e claras, que apesar de introduzirem uma pequena carga adicional de trabalho, irá contribuir para que seja reduzido o tempo gasto em compreender como e por que algo foi feito em um alteração ou correção posterior.

# Vamos colocar a mão na massa?



## Parte 1

- Crie uma pasta chamada **semana1**
- Entre na pasta
- Inicialize o git nesta pasta
- Crie um arquivo chamado **README.md**
- Adicione o arquivo
- Faça um commit
- Faça um push

## Parte 2

- Crie um repositório no GitHub chamado **semana1-GitEGithub**
- Faça um push desse repositório para sua máquina dentro da pasta que criamos
- Abra o arquivo **README.md** no VScode
- Escreva nele algo sobre você e que você quer que os outros saibam
- Faça um commit
- Faça um push

# Dúvidas?



# Tarefa de casa

- Abra o arquivo README.md que criamos em aula no VScode
- Atualize este README com um resumo de algum dos links que estão disponibilizados aqui (ou dos três links se você quiser):
  - **Diferença entre framework e biblioteca**
  - **Como a internet funciona**
  - **O que é algoritmo**
- Neste README precisa conter alguma propriedade Markdown, para saber mais disponibilizei alguns links pra consulta na parte de links úteis no final do repositório da aula.
- Adicione novamente o arquivo
- Faça um commit semântico
- Faça um push das modificações para o seu repositório
- Adicione o link do seu repositório no classroom

