


**Vamos
aprender
Github!**

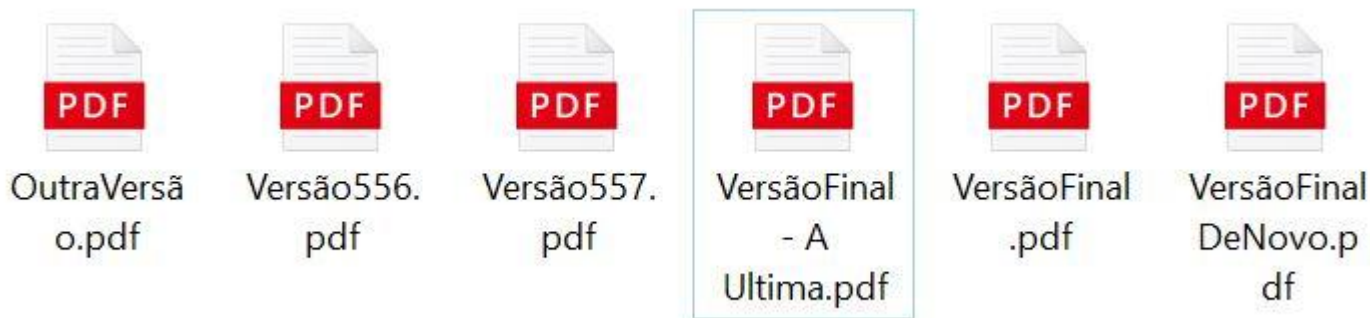


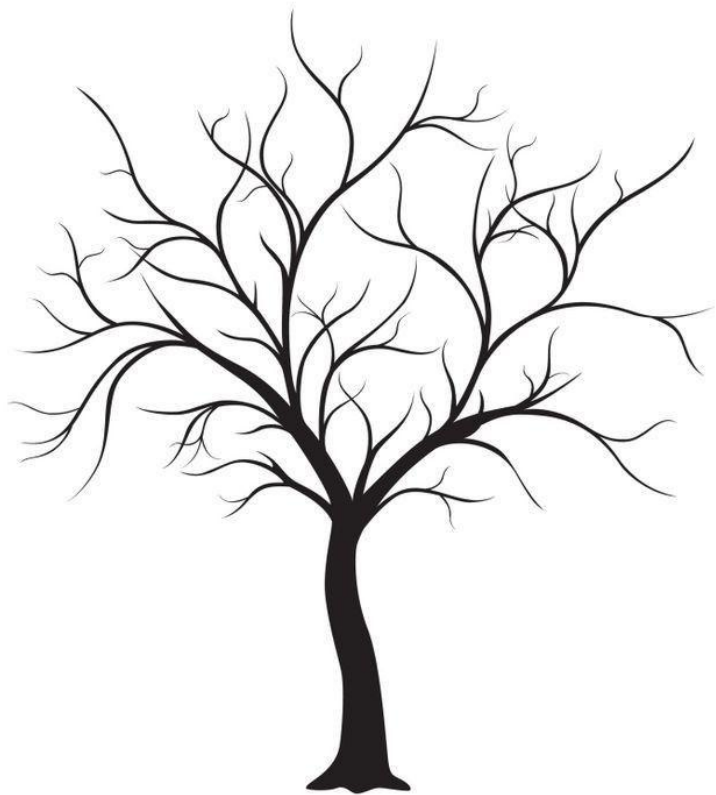
A medium shot of Michael Scott, played by Steve Carell, in his office. He is wearing a dark suit, a light blue shirt, and a striped tie. He has a questioning or slightly exasperated expression on his face, looking off-camera to the right. The background consists of horizontal window blinds. A dark coat is hanging on a rack to the right of the frame.

Why don't you
explain this to me like
I am an eight-year-old?

Por que usamos o Github?

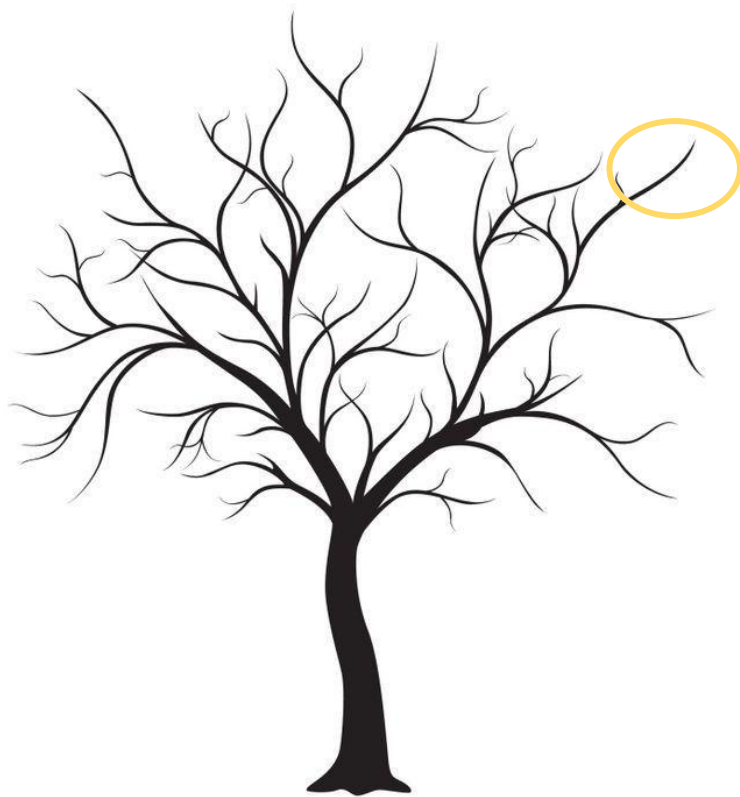
Para programadores trabalharem remotamente sem precisar enviar o código um para o outro diversas vezes por dia





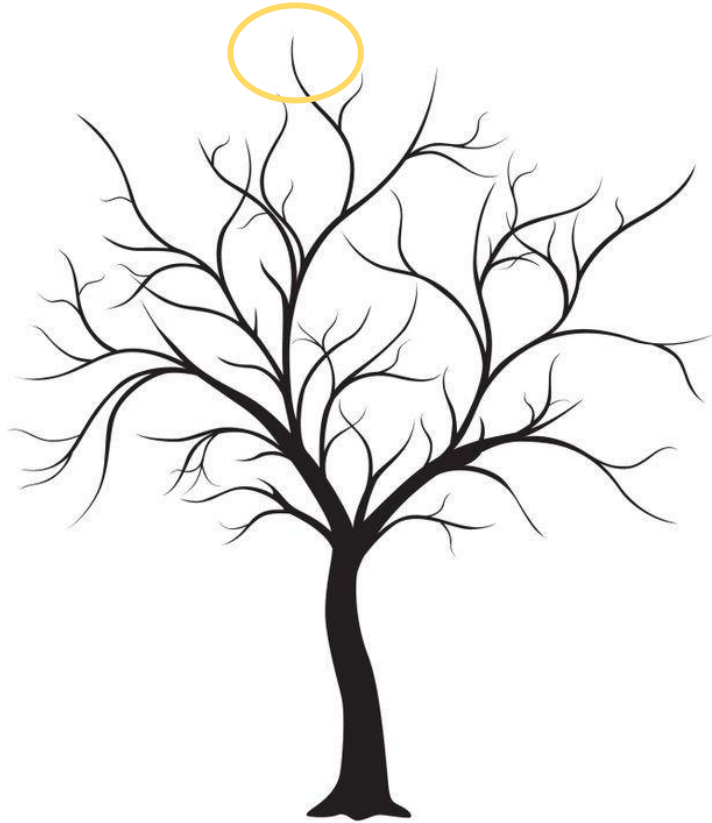
O Github é como uma árvore

Cada galho da árvore
se une ao tronco, que é
ligado a raiz. Vamos
traduzir isso para o
universo da programação!



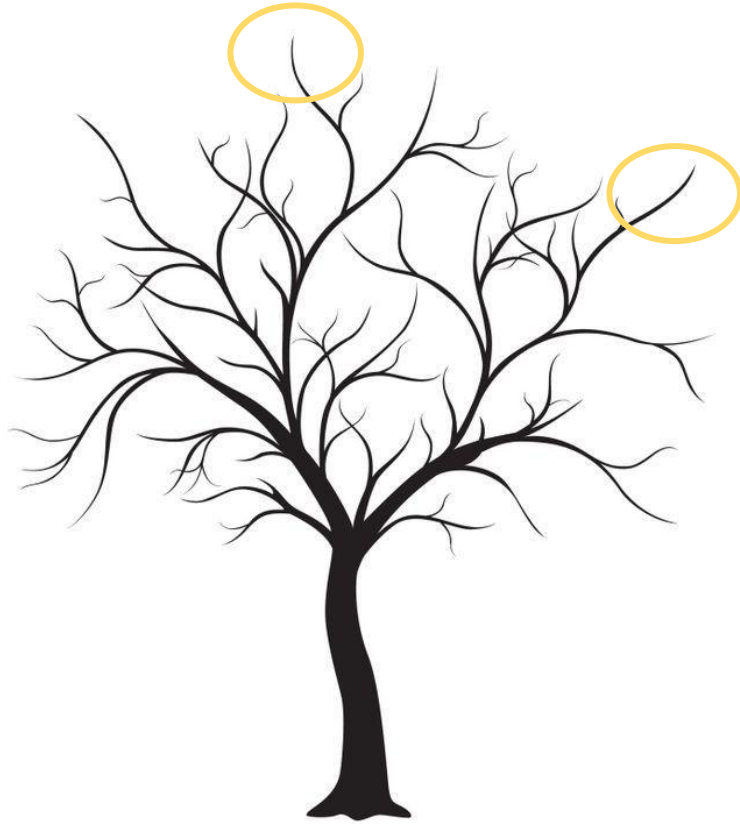
**Branch = o galho
onde você vai
trabalhar**

Exemplo: o programador (A)
vai fazer o sistema de
cadastro de novos usuários
no site. Para isso, ele vai
trabalhar em um galho da
árvore que chamaremos de
branch



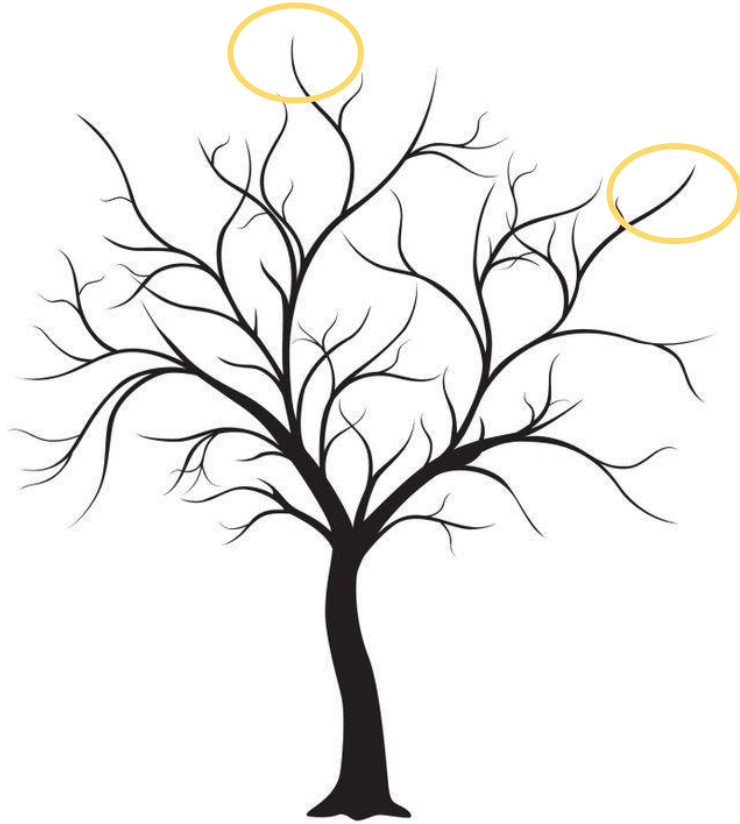
**Branch = o galho
onde você vai
trabalhar**

Enquanto isso, o
programador (B) vai fazer o
sistema de logout do site.
Para isso, ele vai trabalhar
em um outro galho (branch)
da árvore



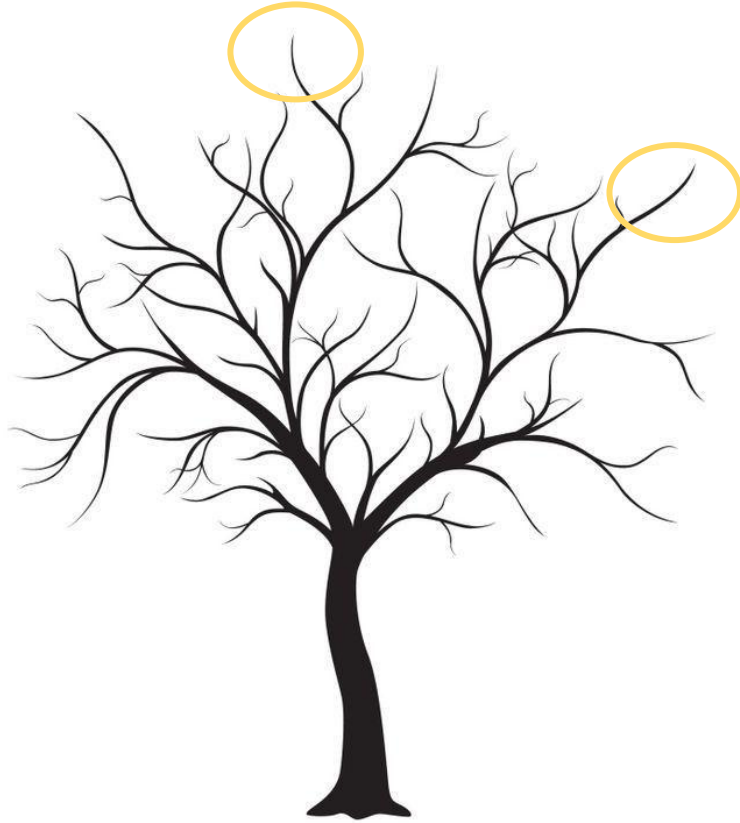
**Branch = o galho
onde você vai
trabalhar**

Perceba que os
programadores (A) e (B)
trabalham em funcionalidades
distintas e, para isso, usam
galhos distintos. Sendo assim
o trabalho de um não interfere
no trabalho do outro!



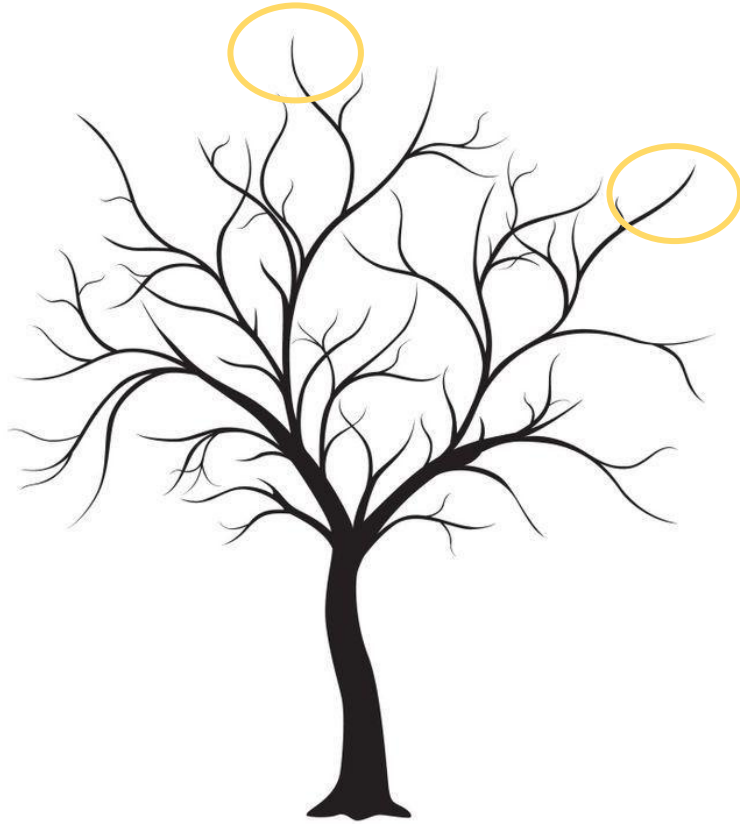
**Cada funcionalidade
é chamada de:
feature**

Então o programador (A)
estava trabalhando na feature
de cadastro, e o programador
(B) estava com a feature de
logout

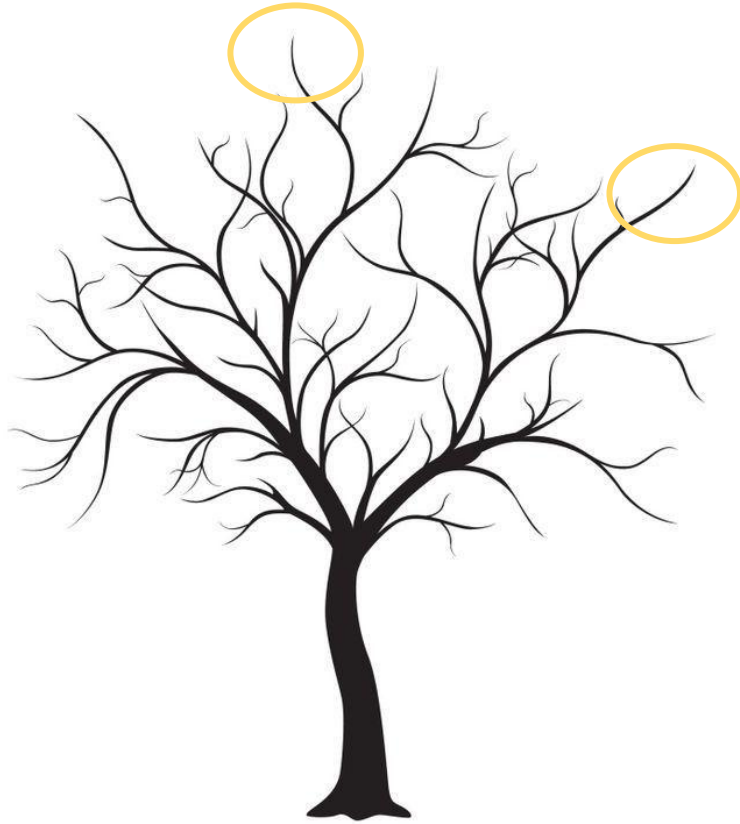


**Então eu desenvolvo
uma *feature* dentro
de uma *branch***

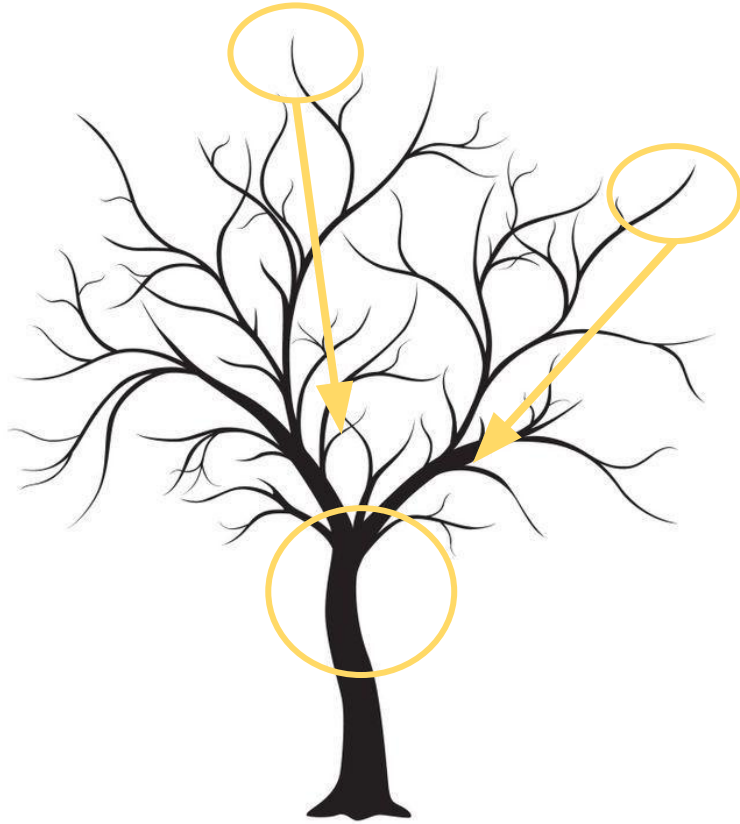
Esses termos parecem
confusos agora, mas fazem
parte do dia a dia dos
programadores. Em breve
vocês se acostumam!



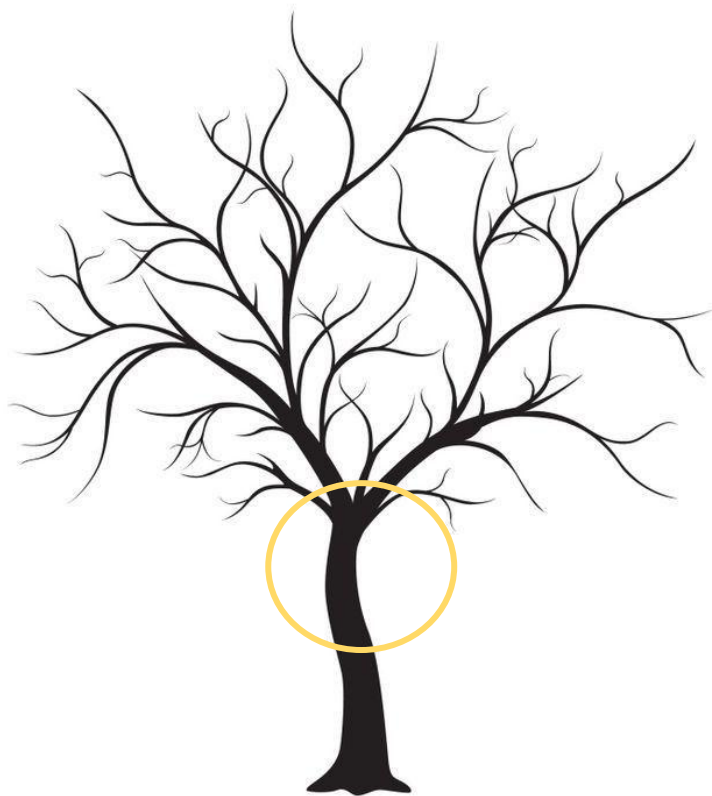
**Pull request =
programador (C),
vê se o que a
gente fez no
galho (branch)
ficou legal?**



**O programador (C)
pode aprovar ou
não seu galho.
Caso ele não
aprove, é
necessário indicar
quais alterações
precisam ser feitas**

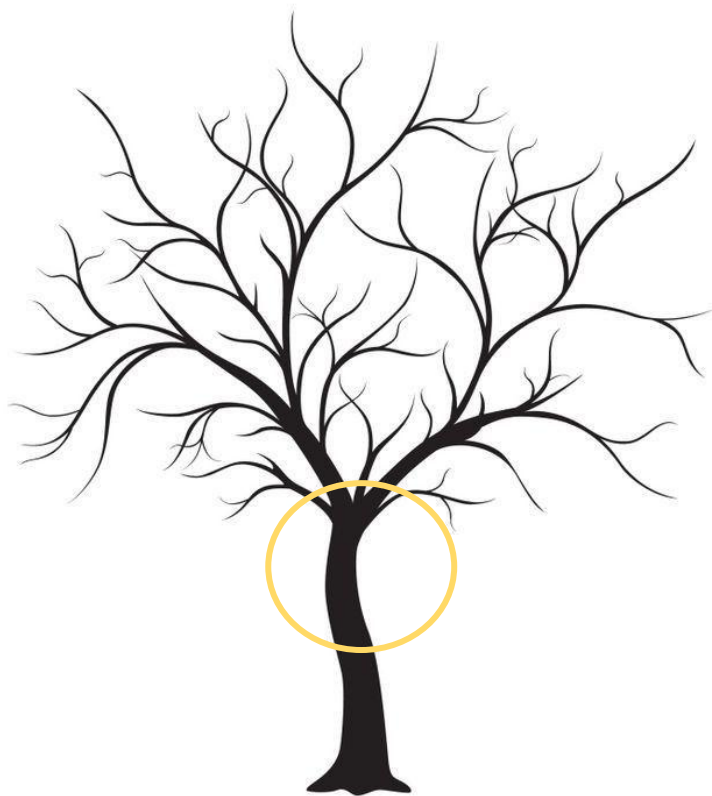


**Quando estiver
tudo certo com seu
galho, ele será
“mergeado” (do
inglês merge, que
significa fundir) na
develop!**



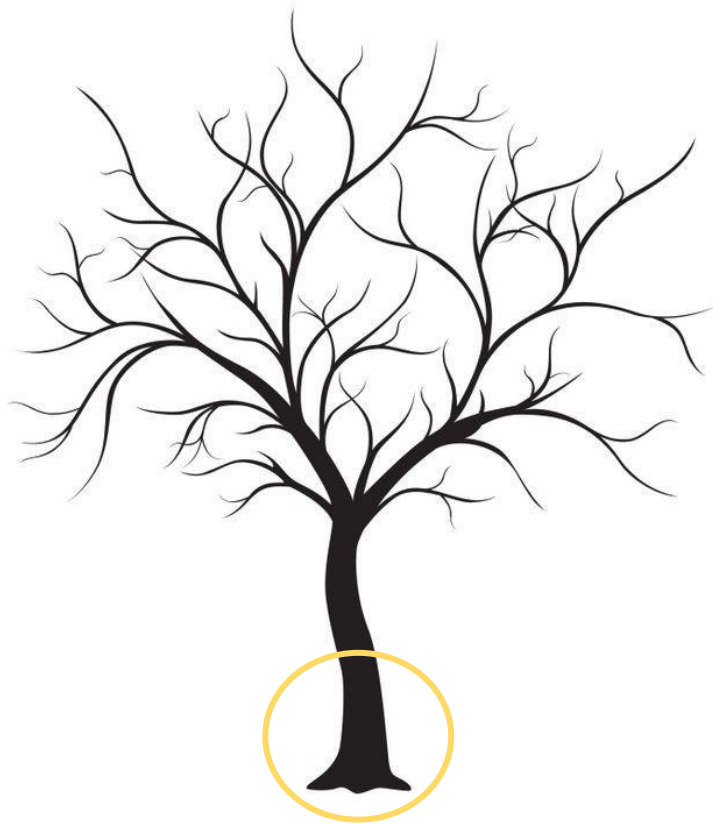
Develop = área de segurança

Aqui unimos todos os trechos de código para conferir se o projeto funciona quando juntamos tudo o que foi feito até então



Develop = área de segurança

Ou seja: será que o sistema de cadastro usa algum parâmetro que interfere no sistema de logout? Tudo funciona corretamente quando os códigos estão juntos?



Raiz = master

Onde guardamos a parte mais estável do código. Ou seja: aqui o site/aplicativo funciona sempre* que o projeto for rodado

Tranquilo, né?

**Agora vamos levar esses
ensinamentos para a prática!**

Master

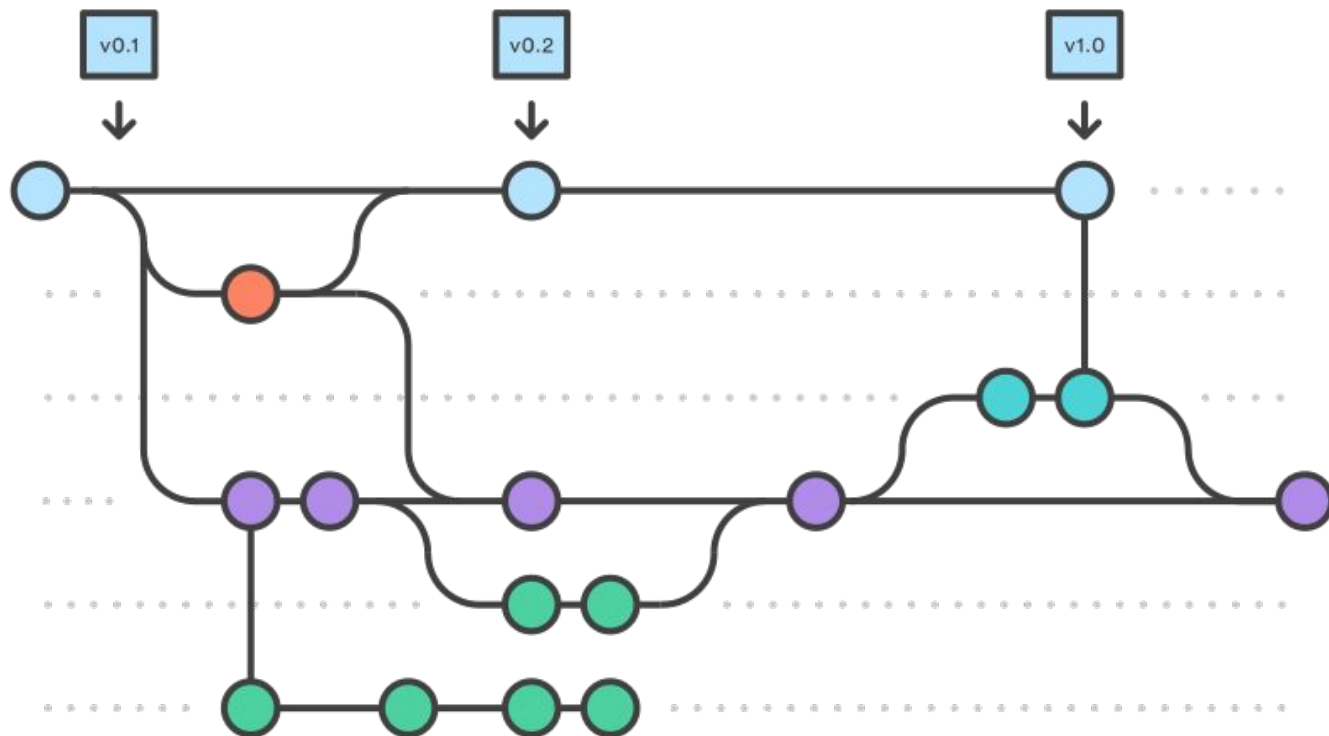
Hotfix

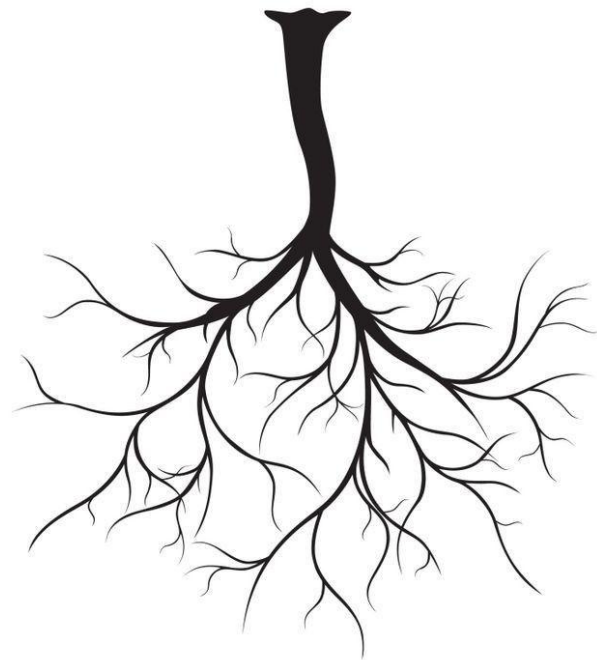
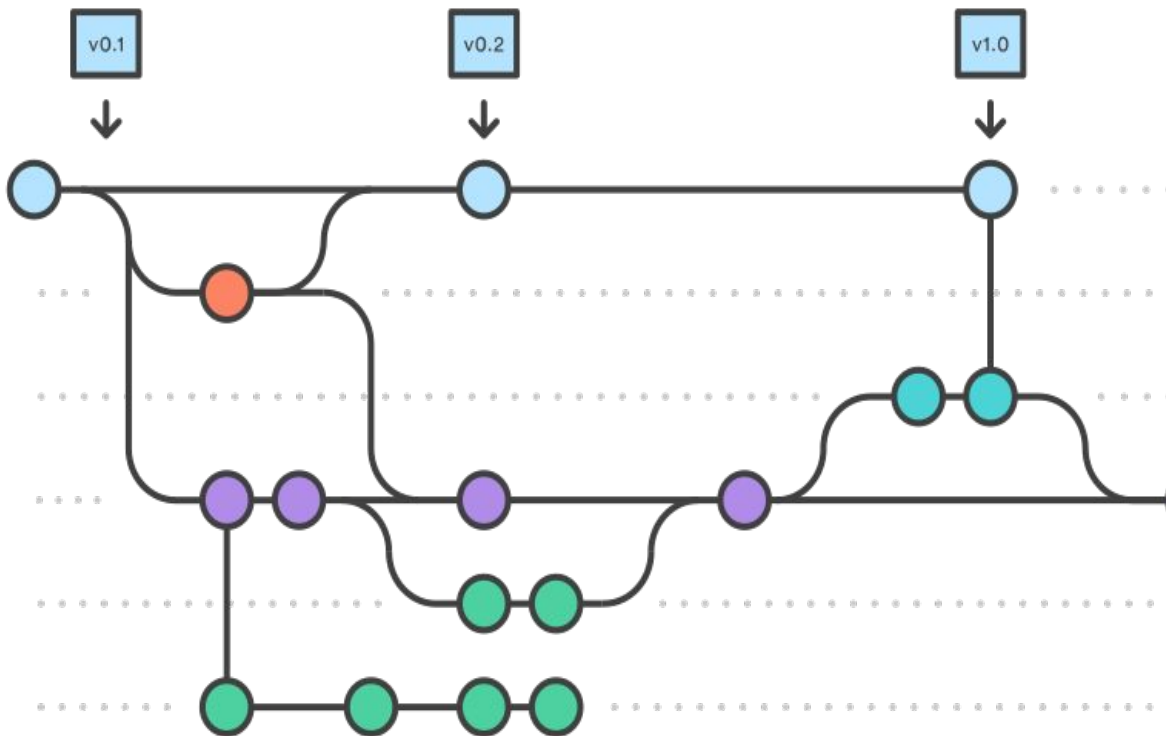
Release

Develop

Feature

Feature





(Antes de começarmos a próxima parte não se esqueça dos comandos básicos do Ubuntu, que estão [aqui](#))

Ok, o projeto está no github.

**Mas como ele vai parar dentro do seu
computador?**

Isso é chamado de clonar o projeto para sua máquina.

(1) Crie uma pasta onde você vai guardar esse projeto no seu computador

(2) Abra o repositório (local do projeto) no git através do link <https://github.com/helenagoulart/repositorioTeste>

(3) Copie o link conforme indicado abaixo:

helenagoulart / repositorioTeste

<> Code ! Issues 🔗 Pull requests ▶ Actions 📁 Projects ! Security 📈 Insights

main 1 branch 0 tags

Go to file

Code

Clone ?

HTTPS GitHub CLI

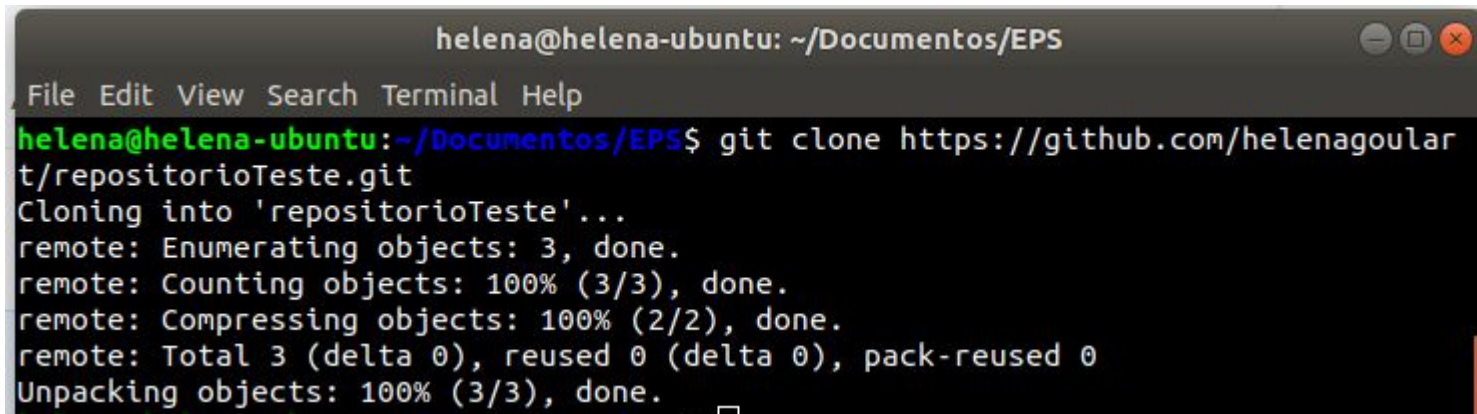
https://github.com/helenagoulart/repositorioTeste.git

Use Git or checkout with SVN using the web URL.

Download ZIP

(4) No terminal, siga os passos indicados

git clone + ctrl + shift + v = clonar o link do repositório e assim trazê-lo para dentro do seu computador

A screenshot of a terminal window titled 'helena@helena-ubuntu: ~/Documentos/EPS'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command 'git clone https://github.com/helenagoulart/repositorioTeste.git' being executed. The output indicates the repository is being cloned into 'repositorioTeste' and shows progress for enumerating, counting, and compressing objects, all completed successfully.

```
helena@helena-ubuntu: ~/Documentos/EPS
File Edit View Search Terminal Help
helena@helena-ubuntu:~/Documentos/EPS$ git clone https://github.com/helenagoulart/repositorioTeste.git
Cloning into 'repositorioTeste'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

Pronto! Agora o seu projeto tá está dentro do seu computador :)

```
helena@helena-ubuntu:~/Documentos/EP5$ ls  
repositorioTeste  
helena@helena-ubuntu:~/Documentos/EP5$ cd repositorioTeste/  
helena@helena-ubuntu:~/Documentos/EP5/repositorioTeste$
```


**Agora vamos enviar uma contribuição
para esse repositório, para simular
como seria enviar um trecho de
código para ele**

(1) Salve um arquivo de texto dentro da pasta do repositório

(2) Escreva no terminal: `git status`

(3) Aparecerão os documentos que estão localmente, ou seja, somente na sua máquina, e os documentos que estão no repositório disponível para todos

git status = mostra os arquivos que estão salvos localmente e os que estão no repositório

```
helela@helela-ubuntu:~/Documentos/EPS/repositorioTeste$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    "Recalrei - Bar\303\265es da Pisadinha"
```

O que está em vermelho se encontra somente na sua máquina, e não no repositório

git add + nome do arquivo = adiciona o arquivo

```
helena@helena-ubuntu:~/Documentos/EPS/repositorioTeste$ git add Recairei\ -\ Barões\ da\ Pisadinha
helena@helena-ubuntu:~/Documentos/EPS/repositorioTeste$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   "Recairei - Bar\303\265es da Pisadinha"
```

Agora que o arquivo está verde, isso significa que você adicionou ele ao repositório

Agora que você adicionou, será necessário commitar.

Neste caso, commitar = postar.

Uma boa prática de programação é enviar junto algum comentário sobre o que está sendo postado

Como fazer um commit

`git commit -m "Sua explicação aqui"`

```
helenah@helenah-ubuntu:~/Documentos/EP5/repositorioTeste$ git commit -m "Included a incredible song"
[main 72e0e35] Included a incredible song
1 file changed, 50 insertions(+)
create mode 100644 "Recarei - Bar\303\265es da Pisadinha"
```

Caso você tenha feito o código em conjunto com um colega, pode commitar em conjunto também, colocando os envolvidos como responsáveis daquele código

Para commitar em conjunto, basta seguir os passos abaixo:

```
$ git commit -m "Refactor usability tests."  
>  
>  
Co-authored-by: name <name@example.com>  
Co-authored-by: another-name <another-name@example.com>"
```


Calma, estamos quase lá.

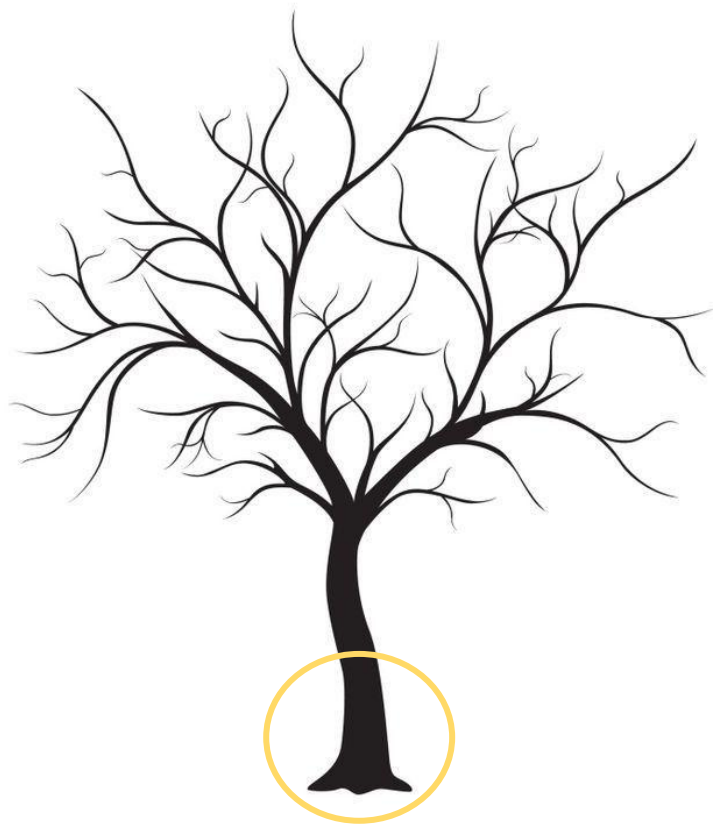
Agora falta enviar de vez o que foi feito! O nome disso é: fazer o push

git push origin + nome da branch = envia para a branch em questão o trecho de código que você fez

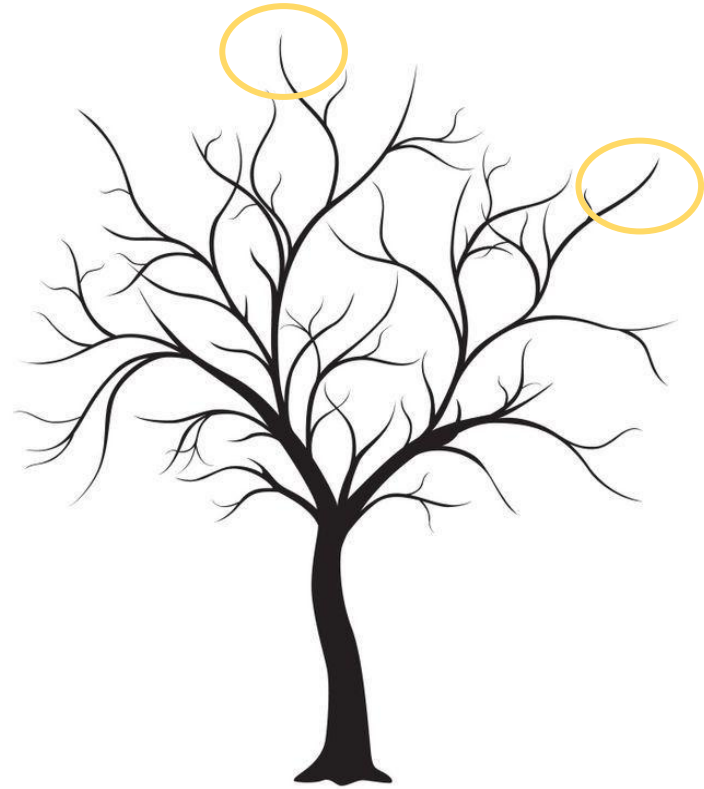
```
helenah@helenah-ubuntu:~/Documentos/EP5/repositorioTeste$ git push origin main
Username for 'https://github.com': helenagoulart
Password for 'https://helenagoulart@github.com':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 561 bytes | 561.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/helenagoulart/repositorioTeste.git
 296ac9a..72e0e35  main -> main
```

Basta preencher seu nome de usuário e senha do github e pronto: você postou um trecho de código na sua branch!

**Nunca
commite
na branch
master.
N-u-n-c-a.
Never. Ever.**



**Como mexer
com branches?**



Para criar uma branch

`git checkout -b + nome da branch`

```
helenahelena-ubuntu:~/Documentos/EPS/repositorioTeste$ git checkout -b BranchHelena  
Switched to a new branch 'BranchHelena'
```

Neste caso criei uma branch chamada: BranchHelena

Assim que você cria uma branch, ele já te coloca dentro dela

Para ver todas as branches existentes no projeto

git branch

```
Switched to a new branch 'BranchHelena'  
helena@helena-ubuntu:~/Documentos/EPS/repositorioTeste$ git branch  
* BranchHelena  
main
```

Neste caso existem duas branches: *main* e *BranchHelena*
Ele está indicando que atualmente estou dentro da BranchHelena

Para trocar de branch

git checkout + nome da branch

```
helenahelena-ubuntu:~/Documentos/EPS/repositorioTeste$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Agora estamos dentro da branch main

Meu colega fez um código e eu quero abri-lo dentro do meu computador. Como lidar?

Isso se chama dar o pull no projeto

git pull origin + nome da branch

Quando fizer isso, você trará todas as atualizações do projeto para o seu computador

É indicado que todas as vezes que for começar a trabalhar, a primeira coisa que deve ser feita é dar o pull

Tarefa de casa

1) Criar uma branch e commitar um arquivo de texto nela

Exigências:

- A branch deve ter seu nome. Exemplo: BranchCibele**
- Não se esqueça de usar os comandos: o git add, git commit e git push neste arquivo**

2) Com um outro arquivo, commitar em conjunto na sua branch em questão. Você pode commitar qualquer arquivo de texto, mas deve usar o co-authored-by para fazer o commit