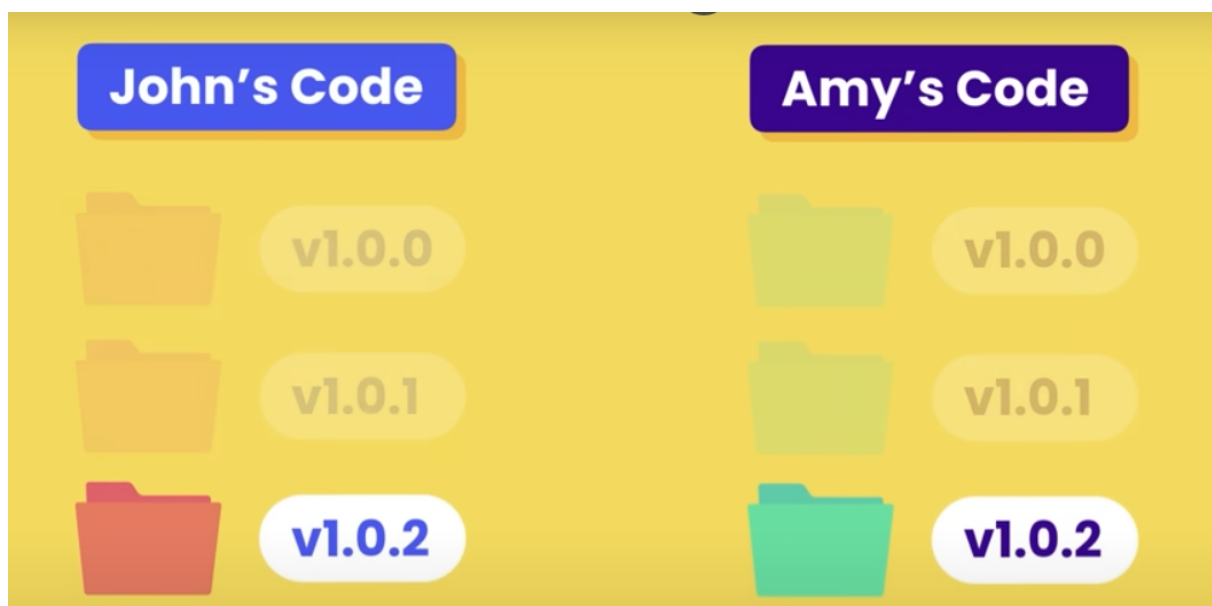


# Introdução

Gratuito e código livre de versionamento de sistema

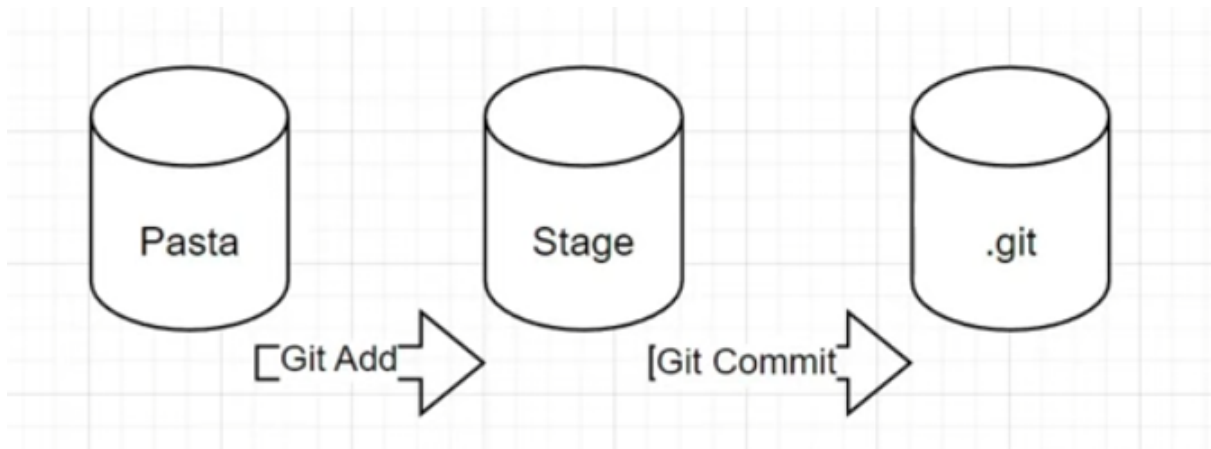
É um sistema de controle de versão que grava mudanças do código ao longo do tempo em um banco de dados especial chamado repositório. É possível olhar o histórico e olhar quem fez as mudanças e o porquê.

Sem o controle de versão, seria necessário guardar as mudanças em arquivos/pasta como v1.0.0 / v1.0.1. Isso seria muito lento, não escalável. O compartilhamento do código mais usual seria através de email ou algum outro mecanismo mais lento.



O motivo de utilizar GIT é porque é gratuito, código livre, muito rápido e escalável.

## Git ADD e STAGE



Ao criar os arquivos usando o comando touch é possível observar os arquivos no explorer:

The screenshot shows a Windows Explorer window with the following table of files:

Nome	Data de modificação	Tipo
arquivo	06/09/2023 17:51	Arquivo
filename	06/09/2023 17:53	Arquivo Fonte
filename	06/09/2023 17:53	Microsoft Edge
filename	06/09/2023 17:53	Documento de
filename.word	06/09/2023 17:54	Arquivo WORD
PrimeiroCommit	01/09/2023 17:54	Arquivo Fonte
README	01/09/2023 21:42	Arquivo Fonte

Below the Explorer window is a terminal window with the following commands and output:

```
MINGW64: e/Guilherme/programacao/gitgihub/projetos/comandos
arquivo
nothing added to commit but untracked files present (use "git add" to track)
guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgihub/p
s (main)
$ git add .
guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgihub/p
s (main)
$ touch filename.txt
guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgihub/p
s (main)
$ touch filename.csv
guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgihub/p
s (main)
$ touch filename.pdf
guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgihub/p
s (main)
$ touch filename.word
```

Já usando o comando git status é possível verificar os arquivos criados em vermelho na tela do bash, eles prontos para serem adicionados na fase stage usando o comando 'add'.

Após o comando 'add .' eles estão na fase stage aguardando o commit para enviar pro repositório

```

guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgithub/projetos/comando
s (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   arquivo

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

guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgithub/projetos/comando
s (main)
$ git add .

guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgithub/projetos/comando
s (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   arquivo
    new file:   filename.csv
    new file:   filename.pdf
    new file:   filename.txt
    new file:   filename.word

guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgithub/projetos/comando
s (main)
$

```

## GIT commit

O commit é uma aplicação de uma mudança no código

```
git commit -m " Adicionando nova funcionalidade"
```

A regra do GIT é todo o commit ter um autor, para que fique registrado no histórico do repositório, para isso é necessário estar logado.

Imagine uma empresa com 200 funcionários, sendo que 1 deles enviou um commit que quebre o sistema, através do histórico, é possível achar a pessoa e qual erro cometeu para rápida correção.

```

guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgithub/projetos/comandos (main)
$ git commit -m " adicionando mais arquivos no projeto"
[main d3b26b1] adicionando mais arquivos no projeto
5 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 arquivo
create mode 100644 filename.csv
create mode 100644 filename.pdf
create mode 100644 filename.txt
create mode 100644 filename.word

guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgithub/projetos/comandos (main)

```

## Git log

mostra o histórico dos commits já realizado por cada pessoa que contribuiu.

```
git log
```

```

guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgithub/projetos/comandos
$ git log
commit d3b26b11820509fd4eb7a4674548d66c841cf540 (HEAD -> main)
Author: Guihtech <guilherme.oliveira.antonio@gmail.com>
Date:   Wed Sep 6 18:09:59 2023 -0300

    adicionando mais arquivos no projeto

commit c816b237fb784e70e92026720d15cea0531637be
Author: Guihtech <guilherme.oliveira.antonio@gmail.com>
Date:   Fri Sep 1 21:42:04 2023 -0300

    first commit

commit 6eac7a3e18ca1328860b20ae8ee44906cb2de253
Author: Guihtech <guilherme.oliveira.antonio@gmail.com>
Date:   Fri Sep 1 21:38:26 2023 -0300

    primeiro

commit 8f3ac8addfa6fce772286abae008efd2f3bf2248
Author: Guihtech <guilherme.oliveira.antonio@gmail.com>
Date:   Fri Sep 1 18:33:53 2023 -0300

    first commit

commit 2951e472d48567a307974eca68cf3e17b298322c
Author: Guihtech <guilherme.oliveira.antonio@gmail.com>
Date:   Fri Sep 1 18:33:25 2023 -0300

    first commit

```

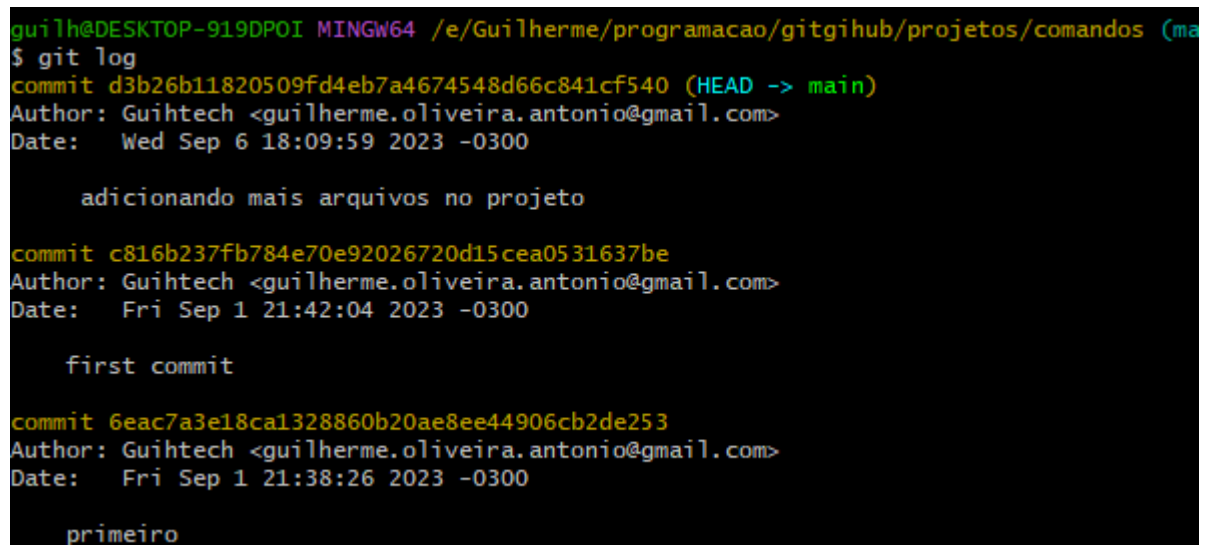
cada commit realizado tem um hash , um identificador único( igual o CPF), no commit mais novo possui um ponteiro chamado “Head → main”. Através dele é

possível navegar na linha do tempo de todos os commits já realizado.

## Git checkout

Comando que permite voltar versões dos commits já realizado.

```
git log
```

A terminal window screenshot showing the output of the 'git log' command. The terminal has a black background with green and yellow text. The output shows three commits in reverse chronological order. The first commit is 'd3b26b11820509fd4eb7a4674548d66c841cf540 (HEAD -> main)' with the message 'adicionando mais arquivos no projeto'. The second commit is 'c816b237fb784e70e92026720d15cea0531637be' with the message 'first commit'. The third commit is '6eac7a3e18ca1328860b20ae8ee44906cb2de253' with the message 'primeiro'. Each commit entry includes the author 'Guihtech <guilherme.oliveira.antonio@gmail.com>' and the date 'Fri Sep 1 21:38:26 2023 -0300' or 'Wed Sep 6 18:09:59 2023 -0300'.

```
guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgithub/projetos/comandos (ma
$ git log
commit d3b26b11820509fd4eb7a4674548d66c841cf540 (HEAD -> main)
Author: Guihtech <guilherme.oliveira.antonio@gmail.com>
Date:   Wed Sep 6 18:09:59 2023 -0300

    adicionando mais arquivos no projeto

commit c816b237fb784e70e92026720d15cea0531637be
Author: Guihtech <guilherme.oliveira.antonio@gmail.com>
Date:   Fri Sep 1 21:42:04 2023 -0300

    first commit

commit 6eac7a3e18ca1328860b20ae8ee44906cb2de253
Author: Guihtech <guilherme.oliveira.antonio@gmail.com>
Date:   Fri Sep 1 21:38:26 2023 -0300

    primeiro
```

Utilizar git checkout [ 6 caracteres iniciais do commit]

```
git checkout c816b2
```

```

guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgithub/projetos/comandos (main)
$ git checkout c816b2
Note: switching to 'c816b2'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at c816b23 first commit

guilh@DESKTOP-919DPOI MINGW64 /e/Guilherme/programacao/gitgithub/projetos/comandos ((c816b23...))
$ git log
commit c816b237fb784e70e92026720d15cea0531637be (HEAD)
Author: Guihtech <guilherme.oliveira.antonio@gmail.com>
Date:   Fri Sep 1 21:42:04 2023 -0300

    first commit

commit 6eac7a3e18ca1328860b20ae8ee44906cb2de253
Author: Guihtech <guilherme.oliveira.antonio@gmail.com>
Date:   Fri Sep 1 21:38:26 2023 -0300

    primeiro

```

nunca dar git checkout sem dar commit do código mais atual, caso contrário, irá perder o código.

Ao dar git checkout é necessário respeitar a linha do tempo, caso utilize o código antigo, evitar ao máximo dar commit para não bagunçar o que já foi feito

Para desfazer o comando, é possível dar git checkout master

```
git checkout master
```