# Aula #12: Visualizando e Arrumando o Último commit do Histórico Local

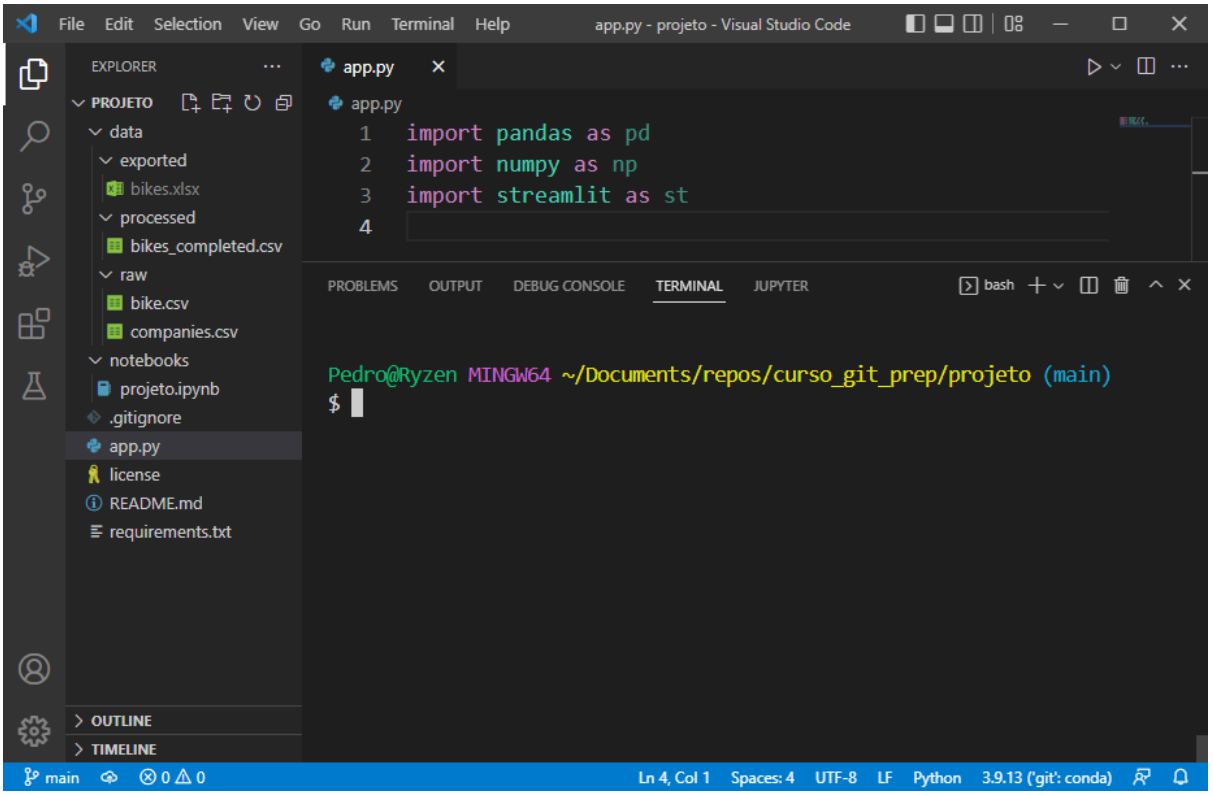
---



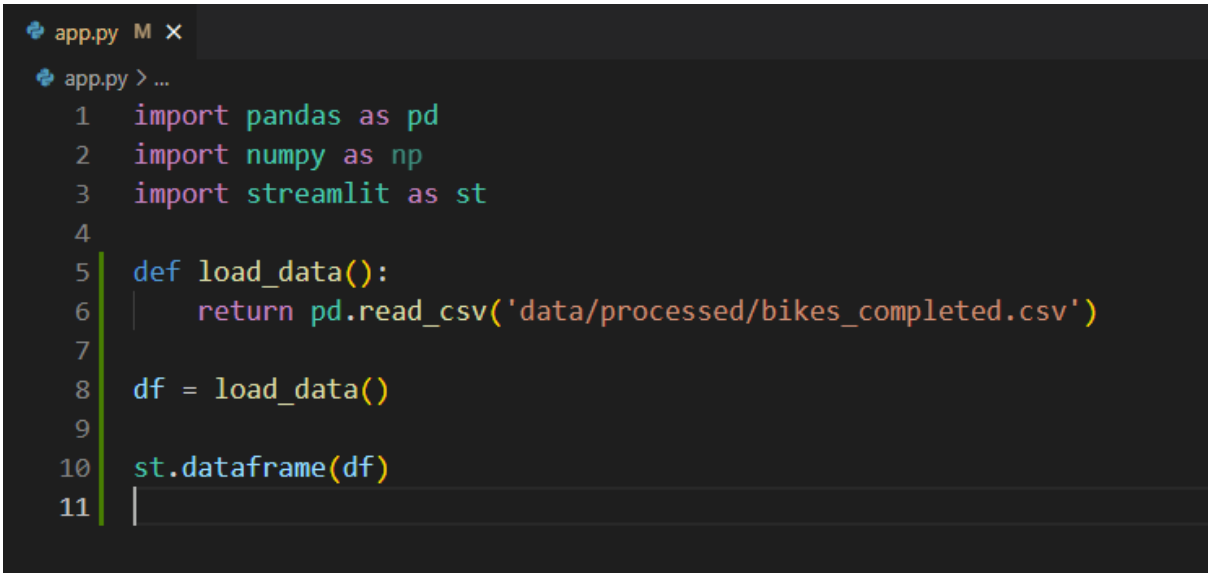
---

## Visualizando commits

Agora que já temos o domínio dos comandos mais básicos, vamos criar mais alguns commits e começar a montar o nosso projeto final. Primeiro, abra o do curso, com toda a estrutura de arquivos de dados, notebooks e Python:




Com o projeto aberto, vamos ajustar o nosso arquivo app.py. Vamos adicionar a função load_csv() para fazer a leitura do nosso arquivo de dados, e criar uma seção que será utilizada para exibir esse resultado dentro do Streamlit:




Com essa alteração feita, vamos salvar o novo estado do nosso arquivo. Ou seja, vamos commitar as nossas alterações com a mensagem “Efetuado carga de dados e apresentado dataframe”:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git add app.py

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git commit -m "Efetuado carga de dados e apresentado dataframe"
[main d3f86fa] Efetuado carga de dados e apresentado dataframe
 1 file changed, 7 insertions(+)
```

Vamos adicionar mais uma seção em nosso arquivo app.py. Vamos adicionar uma nova cláusula no final do arquivo e criar uma função “principal”:

app.py M

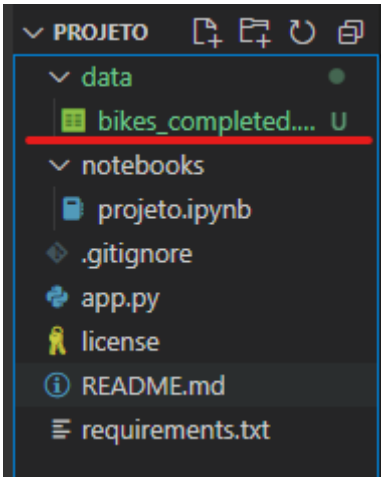
app.py > ...

```python
import pandas as pd
import numpy as np
import streamlit as st

def load_data():
    return pd.read_csv('data/processed/bikes_completed.csv')

def main():
    df = load_data()

    st.dataframe(df)

if __name__ == '__main__':
    main()
```

Vamos commitar essas alterações com a mensagem “Adicionado funcao principal”:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git add app.py

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git commit -m "Adicionado funcao principal"
[main 9834a5a] Adicionado funcao principal
 1 file changed, 6 insertions(+), 2 deletions(-)
```

Vamos arrumar a estrutura de diretórios do projeto. Vamos excluir as pastas exported, processed e raw, e vamos deixar somente o arquivo bikes_completed.csv:




Vamos commitar essas alterações com a mensagem “Ajustado estura de patas”, com o erro de português mesmo:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    data/processed/bikes_completed.csv
        deleted:    data/raw/bike.csv
        deleted:    data/raw/companies.csv

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

no changes added to commit (use "git add" and/or "git commit -a")
```

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git add .
warning: in the working copy of 'data/bikes_completed.csv', CRLF will be replaced by LF the n
ext time Git touches it

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git commit -m "Ajustado estura de patas"
[main 8121717] Ajustado estura de patas
 3 files changed, 1167 deletions(-)
 rename data/{processed => }/bikes_completed.csv (100%)
 delete mode 100644 data/raw/bike.csv
 delete mode 100644 data/raw/companies.csv
```

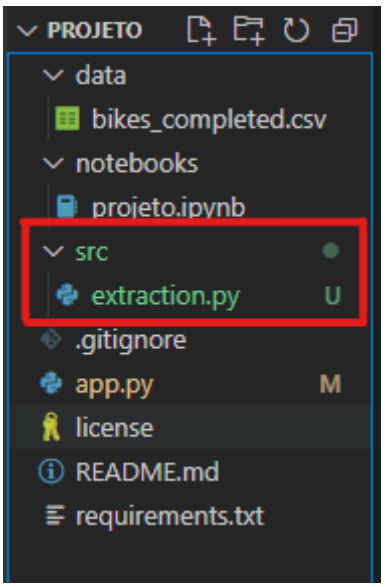
Vamos fazer a nossa última alteração no arquivo app.py:

app.py M

app.py > ...

```python
import streamlit as st
from src.extraction import load_data

st.set_page_config(layout="wide")

def main():
    df = load_data()

    st.dataframe(df)

if __name__ == '__main__':
    main()
```

Junto com a alteração do arquivo app.py, vamos criar um novo arquivo chamado extraction, que deverá ficar dentro de uma pasta chamada src, que deve ficar na raiz do projeto:




E o conteúdo do arquivo extraction é função de carregamento do arquivo csv:


Vamos commitar essas alterações com a mensagem “Criado arquivo de extracao e ajustado arquivo prinpial”, com o erro de português:

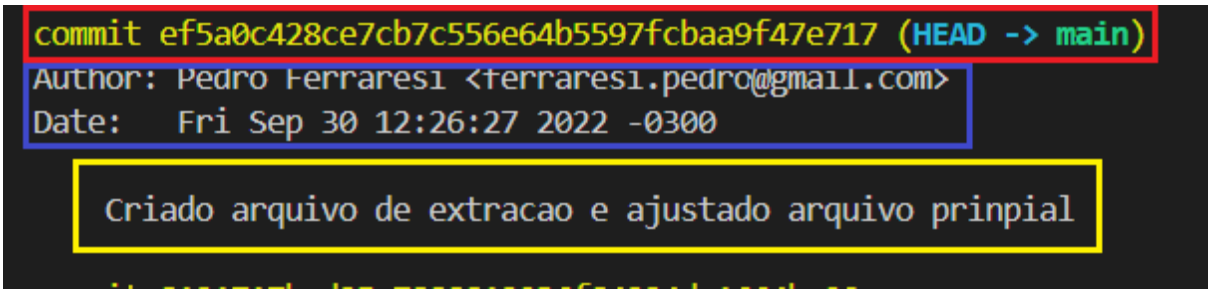
```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   app.py

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

no changes added to commit (use "git add" and/or "git commit -a")
```

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git add .
warning: in the working copy of 'src/extraction.py', CRLF will be replaced by LF the next tim
e Git touches it

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git commit -m "Criado arquivo de extracao e ajustado arquivo prinpial"
[main ef5a0c4] Criado arquivo de extracao e ajustado arquivo prinpial
 2 files changed, 6 insertions(+), 4 deletions(-)
 create mode 100644 src/extraction.py
```

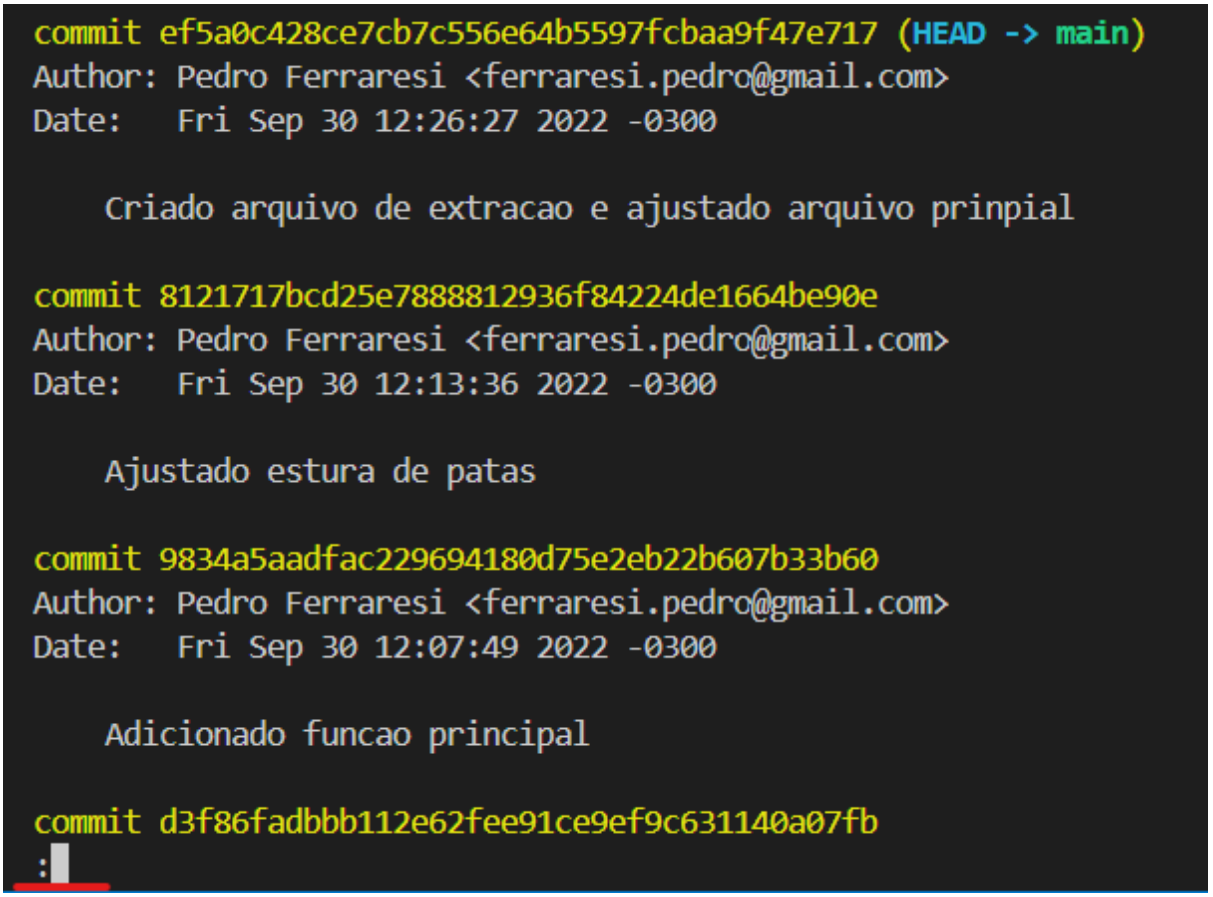
Agora que temos várias commits feitos no repositório, como podemos verificar o que foi feito? É aí que entra o comando git log. O comando git log nos permite visualizar o histórico de alterações feitas dentro do repositório local:

```
commit ef5a0c428ce7cb7c556e64b5597fcbaa9f47e717 (HEAD -> main)
Author: Pedro Ferraresi <ferraresi.pedro@gmail.com>
Date:   Fri Sep 30 12:26:27 2022 -0300

    Criado arquivo de extracao e ajustado arquivo prinpial

commit 8121717bcd25e7888812936f84224de1664be90e
Author: Pedro Ferraresi <ferraresi.pedro@gmail.com>
Date:   Fri Sep 30 12:13:36 2022 -0300

    Ajustado estura de patas

commit 9834a5aadfac229694180d75e2eb22b607b33b60
Author: Pedro Ferraresi <ferraresi.pedro@gmail.com>
Date:   Fri Sep 30 12:07:49 2022 -0300

    Adicionado funcao principal

commit d3f86fadbbb112e62fee91ce9ef9c631140a07fb
:
```

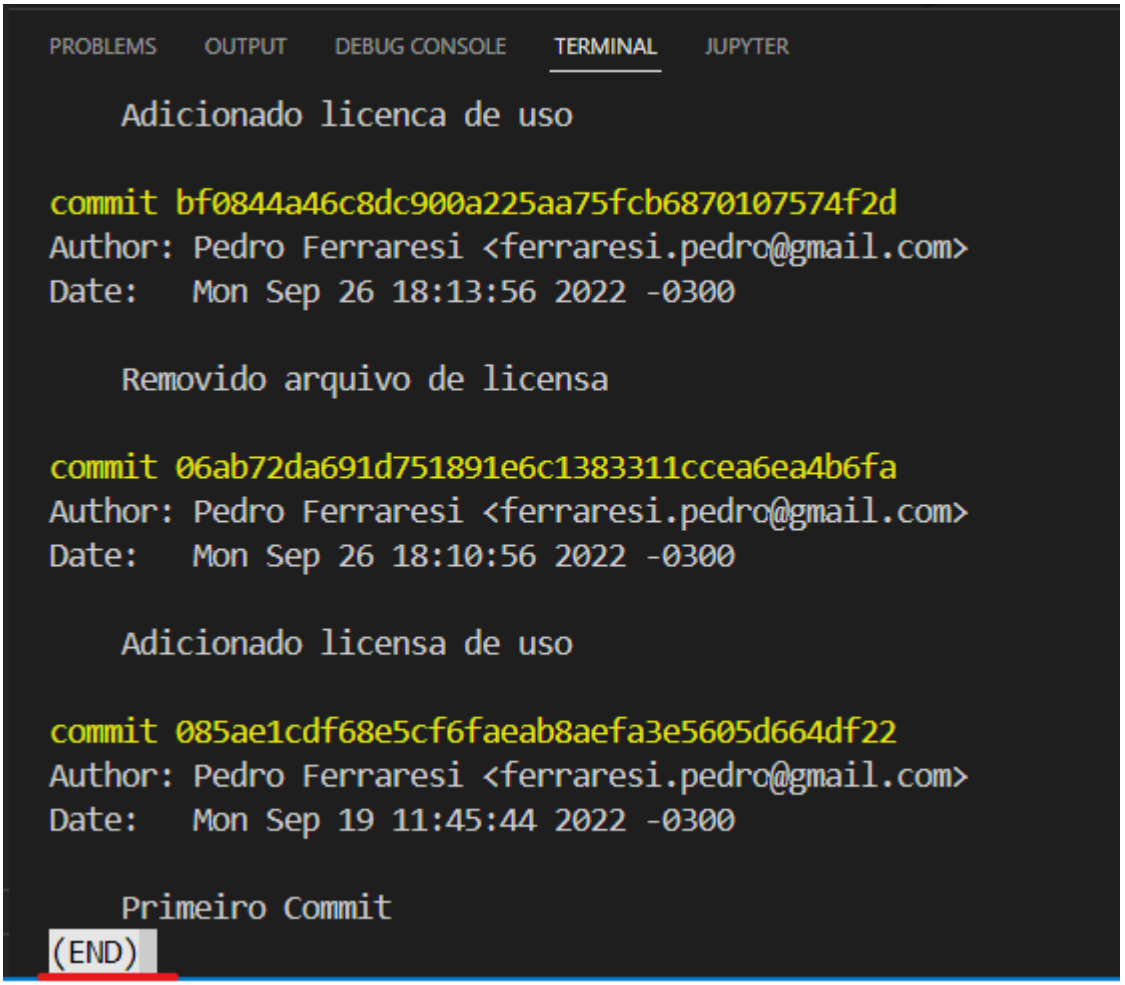
Se pegarmos somente o último commit feito, e observarmos a estrutura dele, podemos verificar diversas informações a respeito desse commit:




O SHA-1 do commit, que é o ID do commit, está marcado em vermelho. O autor e a data do commit estão marcados em azul e a mensagem do commit está marcada em amarelo.




Outro ponto importante, se formos ao final do terminal, é que existe o caractere de dois pontos, conforme imagem acima. Isso significa que existem mais commits para serem exibidos, além dos 4 que estão na tela. Para acessar essas informações, basta clicar na seta para baixo no teclado. Quando chegamos ao fim do histórico de commits, é exibido a palavra END




Para voltarmos a ter o controle do terminal, basta apertar a tecla Q (de Quit) do teclado. Porém, existe uma forma mais simples de verificarmos esse histórico: Utilizando o parâmetro --oneline junto ao comando git log. Esse parâmetro trará somente o SHA-1 do commit, na forma reduzida, e a sua mensagem:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git log --oneline
ef5a0c4 (HEAD -> main) Criado arquivo de extracao e ajustado arquivo prinpial
8121717 Ajustado estura de patas
9834a5a Adicionado funcao principal
d3f86fa Efetuado carga de dados e apresentado dataframe
f1dc7f9 Adicionado Biblioteca Streamlit
ea2b047 Arquivo app.py adicionado
53f930a Removido o arquivo app.py
35faa5d Adicionado bibliotecas pandas e numpy
178c9cb Adicionado arquivo app.py e rodado arquivo do projeto novamente
ca6e4b9 Adicionado licenca de uso
bf0844a Removido arquivo de licensa
06ab72d Adicionado licensa de uso
085ae1c Primeiro Commit
```

Essa maneira é mais rápida e simples de verificarmos o histórico de commits.

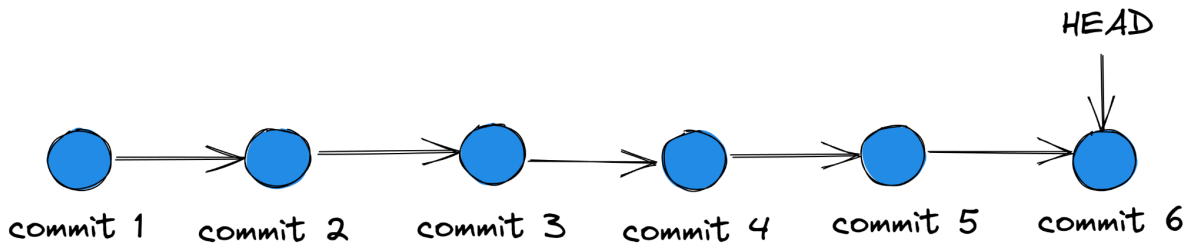
## O ponteiro HEAD do Git

Um conceito importante de entendermos é que esse histórico de commits nos conta a história do nosso projeto. Ou seja, quando lemos o histórico de commits, temos que entender o que foi feito ao longo do projeto. E esses commits podem ser

Além disso, existe um outro conceito muito importante: O ponteiro do Git. Esse ponteiro é representado pelo HEAD que vemos na saída do comando log

```
commit ef5a0c428ce7cb7c556e64b5597fcbaa9f47e717 (HEAD -> main)
Author: Pedro Ferraresi <ferraresi.pedro@gmail.com>
Date:   Fri Sep 30 12:26:27 2022 -0300

    Criado arquivo de extracao e ajustado arquivo prinpial
```

Para simplificar, observe a imagem abaixo:




Cada bolinha azul é um commit feito, e o HEAD, sempre irá apontar para o último commit do histórico de commits. O que ocorre é que o Git utiliza esse ponteiro para se mover pelos commits quando precisamos alterar as informações do nosso histórico, como desfazer commits por exemplo.

Por enquanto, vamos nos ater somente em entender o conceito do HEAD: Que é um ponteiro que indica em qual commit estamos dentro do histórico de commits. Todas as outras funcionalidades desse ponteiro, iremos verificar nas outras aulas.

# Alterando o Último Commit

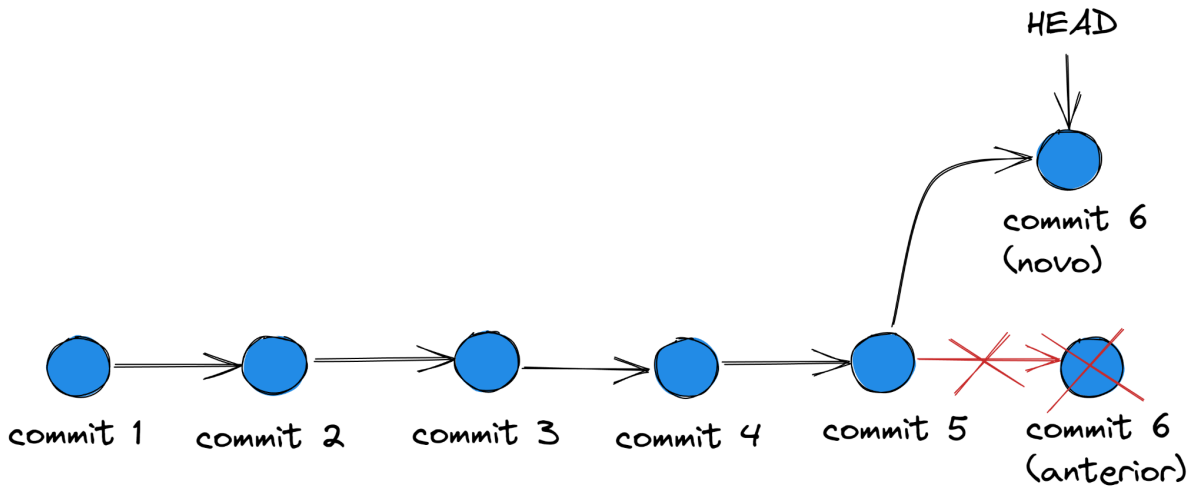
Vamos observar novamente o nosso histórico de commits:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git log --oneline
ef5a0c4 (HEAD -> main) Criado arquivo de extracao e ajustado arquivo prinpial
8121717 Ajustado estura de patas
9834a5a Adicionado funcao principal
d3f86fa Efetuado carga de dados e apresentado dataframe
f1dc7f9 Adicionado Biblioteca Streamlit
ea2b047 Arquivo app.py adicionado
53f930a Removido o arquivo app.py
35faa5d Adicionado bibliotecas pandas e numpy
178c9cb Adicionado arquivo app.py e rodado arquivo do projeto novamente
ca6e4b9 Adicionado licenca de uso
bf0844a Removido arquivo de licensa
06ab72d Adicionado licensa de uso
085ae1c Primeiro Commit
```

Como podemos observar, o último commit do nosso histórico está com erro em sua mensagem. Para arrumar, podemos podemos utilizar o comando git commit -m “Mensagem”, passando o parâmetro --amend:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git commit -m "Criado arquivo de extracao e ajustado arquivo principal" --amend
[main 31b8977] Criado arquivo de extracao e ajustado arquivo principal
 Date: Fri Sep 30 12:26:27 2022 -0300
 2 files changed, 6 insertions(+), 4 deletions(-)
 create mode 100644 src/extraction.py
```

Quando fazemos isso, o Git refaz o último commit e atualiza todo o histórico. Observe que agora, além da mensagem estar correta, o SHA-1 do commit também foi alterado.




Só que o comando git commit --amend não arruma somente a mensagem! Podemos adicionar novas alterações e ajustes a esse último commit. Por exemplo, digamos que encontramos um erro precisamos renomear a variável df para df_raw. Para não termos que gerar um novo commit somente para esse ajuste, podemos utilizar o comando git commit --amend para isso. Vamos fazer essa alteração:

app.py M

app.py > ...

```python
import streamlit as st
from src.extraction import load_data

st.set_page_config(layout="wide")

def main():
    df_raw = load_data()

    st.dataframe(df_raw)

if __name__ == '__main__':
    main()
```

Observe que foi realizado a alteração do arquivo e que essa alteração está dentro da nossa Working Directory:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   app.py

no changes added to commit (use "git add" and/or "git commit -a")
```

Vamos fazer o processo de um commit normal e agora passar o comando git commit --amend:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git add app.py

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git commit -m "Criado arquivo de extracao e ajustado arquivo principal" --amend
[main 97e2251] Criado arquivo de extracao e ajustado arquivo principal
 Date: Fri Sep 30 12:26:27 2022 -0300
 2 files changed, 8 insertions(+), 6 deletions(-)
 create mode 100644 src/extraction.py
```

Observe que foi gerado um novo commit com as alterações que fizemos. E se olharmos o nosso histórico, ainda teremos a mesma quantidade de commits anteriores:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git log --oneline
97e2251 (HEAD -> main) Criado arquivo de extracao e ajustado arquivo principal
8121717 Ajustado estura de patas
9834a5a Adicionado funcao principal
d3f86fa Efetuado carga de dados e apresentado dataframe
f1dc7f9 Adicionado Biblioteca Streamlit
ea2b047 Arquivo app.py adicionado
53f930a Removido o arquivo app.py
35faa5d Adicionado bibliotecas pandas e numpy
178c9cb Adicionado arquivo app.py e rodado arquivo do projeto novamente
ca6e4b9 Adicionado licenca de uso
bf0844a Removido arquivo de licensa
06ab72d Adicionado licensa de uso
085ae1c Primeiro Commit
```

# Exercícios

Link para o formulário de exercícios de fixação de conteúdo: Exercícios

# Próxima Aula

Na próxima aula, veremos como arrumar o commit anterior, que foram as alterações feitas na estrutura de pastas e que está com a mensagem errada.

# Fontes e Links Complementares

Livro Pro Git - Capítulo 2.2 - Salvando alterações no Git

Livro Pro Git - Capítulo 2.3 - Visualizando o Histórico de Commits

Livro Pro Git - Capítulo 2.4 - Desfazendo coisas no Git