

# Aula #9: Ciclo de vida de um arquivo versionado

---

<b>Áreas de Versionamento</b>	<b>1</b>
Untracked	2
Modified	3
Staged	6
Unmodified	7
<b>Exercícios</b>	<b>15</b>
<b>Próxima Aula</b>	<b>15</b>
<b>Fontes e Links Complementares</b>	<b>15</b>

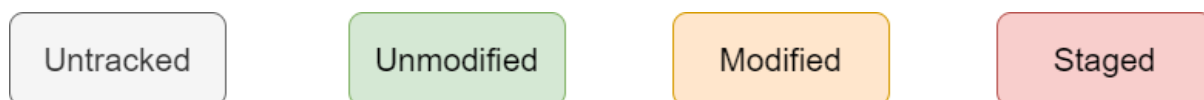
---

## Áreas de Versionamento

Quando versionamos um arquivo, estamos armazenando todas as modificações que são feitas nesse arquivo ao longo do tempo. Ou seja, todas as vezes em que o arquivo for alterado, será gerado um *snapshot* do novo estado dele, que será armazenado no repositório Git.

Porém, o Git possui um mecanismo ou forma de fazer esse controle, que é o ciclo de vida que um arquivo versionado possui. Observe a imagem abaixo:

Existem 4 principais estado em que um arquivo pode estar:



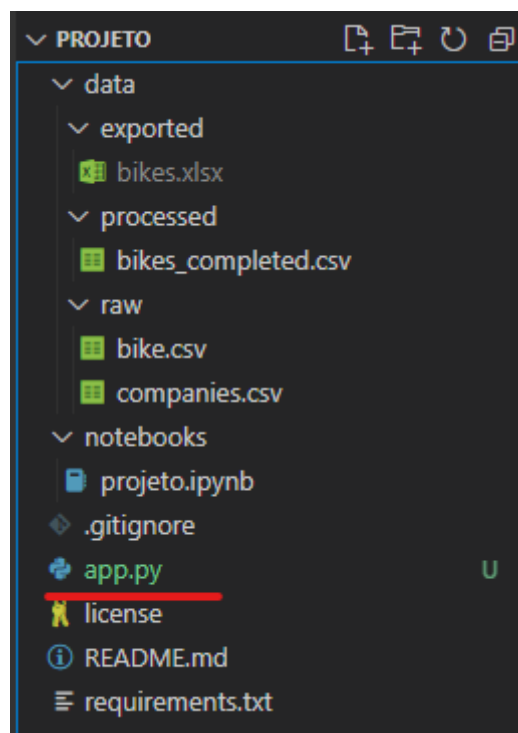
- Untracked
- Unmodified
- Modified
- Staged

Vamos observar com mais detalhes esses 4 estados.

## Untracked

Um arquivo que está com o estado marcado como **Untracked** é todo arquivo novo que é adicionado ao diretório, ou pasta, do projeto mas que nunca teve o seu estado salvo ainda pelo Git. Ou seja, é todo arquivo que adicionamos no projeto mas que ainda não tenha sido gerada uma versão desse arquivo pelo Git.

Para exemplificar melhor, vamos abrir o nosso projeto e criar um novo arquivo chamado **app.py** em nosso projeto.



Com esse arquivo criado, execute o comando **git status** no terminal para verificarmos o estado do nosso repositório Git:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    app.py

nothing added to commit but untracked files present (use "git add" to track)
```

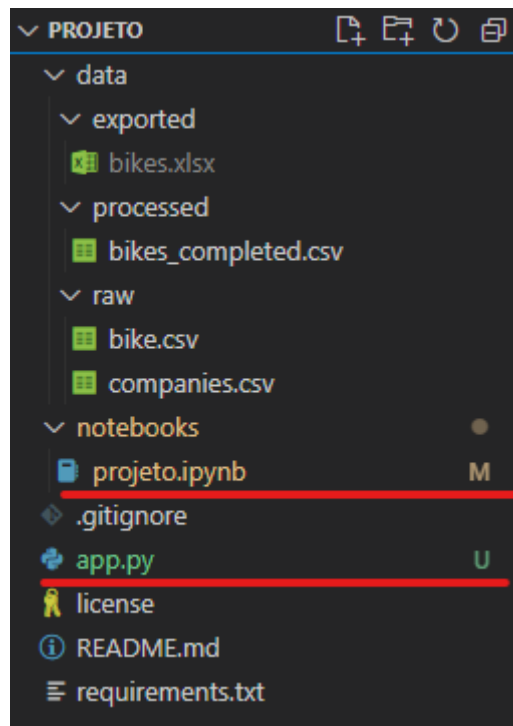
Observe que o arquivo recém criado está dentro da seção de arquivos não rastreados do Git. Ou seja, dentro da seção **Untracked**. Dessa forma, todo o arquivo que é recém adicionado ao projeto, mas ainda não possui um commit para salvar o seu estado, sempre ficará marcado como arquivo **Untracked**.

Ainda não iremos commitar esse arquivo. Primeiro, vamos demonstrar a diferença entre as seções de **Untracked** e **Modified**.

## Modified

A área de **Modified** é a área onde todos os arquivos que já estão sendo rastreados entram quando sofrem alguma alteração. Ou seja, quando temos um arquivo já rastreado pelo Git (que já possui um commit) que sofre alguma alteração, esse arquivo entra na área de **Modified**.

Para exemplificar, abra o arquivo notebook do projeto, execute todas as células e depois salve o arquivo:



Observe que, diferente do arquivo `app.py`, que está dentro da área de **Untracked**, o arquivo `projeto.ipynb` está com uma cor diferente e uma letra diferente na frente de seu nome. A letra **U**, em verde, representa um arquivo que está dentro da área de arquivo **Untracked** (daí o **U** maiúsculo), enquanto a letra **M**, em laranja, representa que o arquivo está dentro da área de **Modified** (daí a letra **M** maiúscula).

Podemos verificar isso utilizando o comando `git status` novamente, para ver o estado do nosso repositório Git:

```
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:   notebooks/projeto.ipynb

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

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

Observe que agora temos duas seções dentro do repositório Git: Uma dizendo que há alterações não marcadas (**not staged**), informando que há um arquivo modificado, e uma outra seção dizendo que temos arquivos não rastreados (**untracked**), ou seja, que ainda não possuem o seu estado salvo com um commit em nosso repositório.

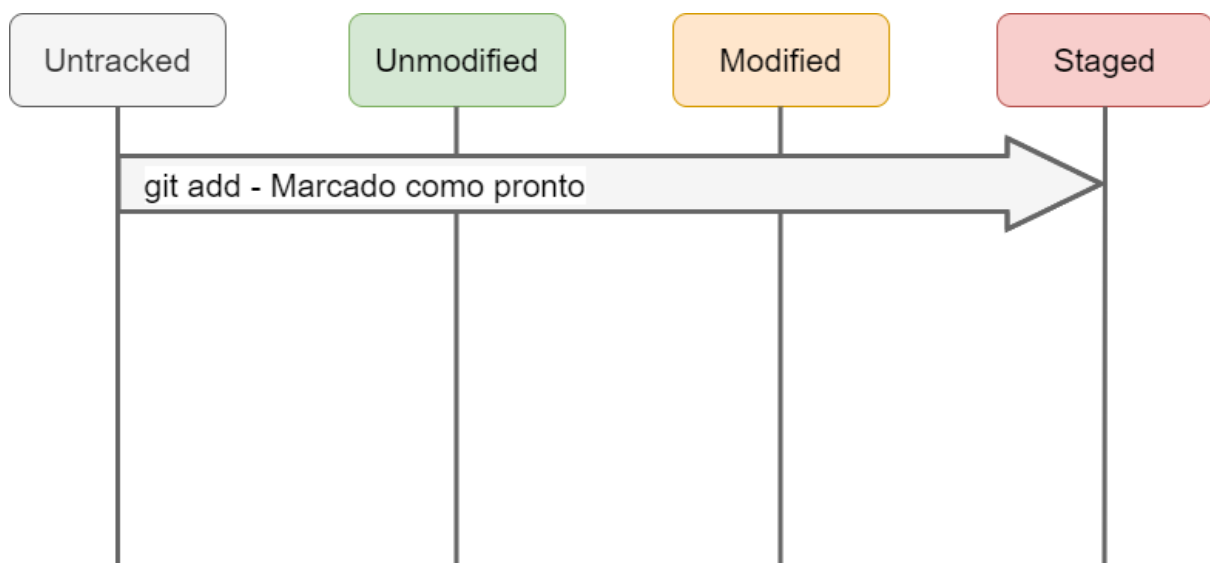
Agora, vamos marcar ambos os arquivos como prontos para demonstrarmos a seção **Staged**. Para marcar um arquivo pronto para ter seu estado salvo com um commit, utilizamos o comando **git add**. Vamos marcar primeiro o arquivo **app.py**:

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

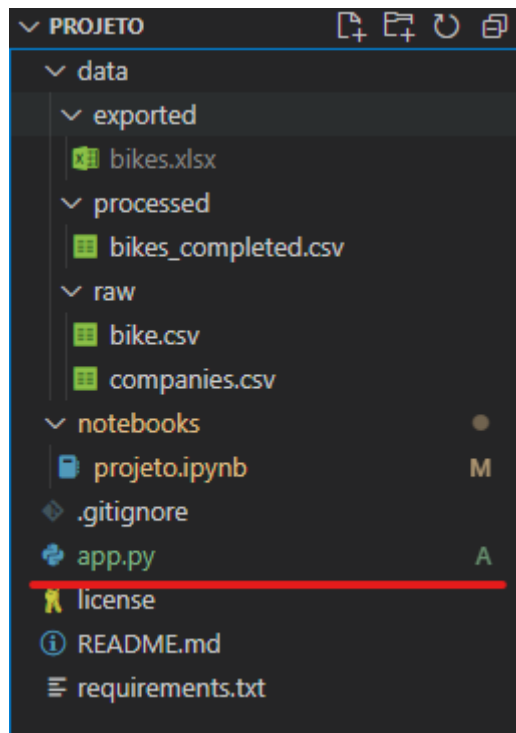
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   app.py

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:   notebooks/projeto.ipynb
```

Observe que, uma vez que marcamos o arquivo como pronto utilizando o comando `git add app.py`, ele saiu da seção **Untracked** e foi para a seção de **Staged**.



Se observarmos dentro da IDE VS Code, podemos ver que a letra do arquivo mudou de **U** para **A**:



O **A** representa que o arquivo foi Adicionado (ou **Added**, daí o **A** maiúsculo) à área de **Staged**.

## Staged

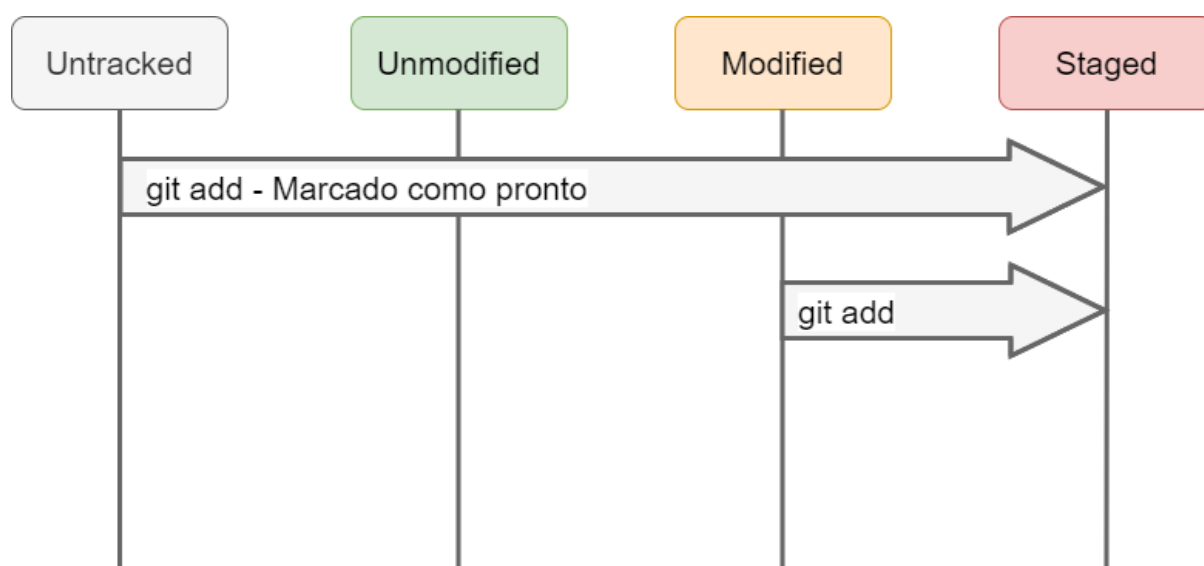
A área de **Staged** é a área onde todos os arquivos prontos para terem seu estado salvo com um commit ficam. Ou seja, sempre que um arquivo foi alterado e está pronto e funcionando, adicionamos ele na área de **Staged** com o comando **git add <nome\_arquivo>** para informarmos ao Git que esse arquivo está pronto para ter o seu estado salvo.

Vamos fazer o mesmo processo de adicionar arquivo para a área de **Staged**, mas agora com o arquivo notebook **projeto.ipynb**. Para isso, basta utilizar o comando **git add notebooks/projeto.ipynb** no terminal. Observe que devemos sempre respeitar a estrutura de pastas do nosso projeto. Ou seja, caso o arquivo que desejamos marcar como pronto está dentro de uma pasta, temos que utilizar o comando relativo até esse arquivo:

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

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   app.py
    modified:   notebooks/projeto.ipynb
```

Observe que, mesmo ambos os arquivos estando dentro da área de **Staged**, um está marcado como arquivo modificado, enquanto o outro está marcado como um novo arquivo.



Agora que ambos os arquivos estão marcados como prontos, vamos commitar as nossas alterações para explicar a seção **Unmodified**.

## Unmodified

A seção **Unmodified** é a seção onde todos os arquivos que possuem alguma versão salva dentro do repositório Git estão. Ou seja, sempre quando commitamos



um arquivo, ele vai para essa seção do repositório e só sai dela caso sofra alguma modificação.

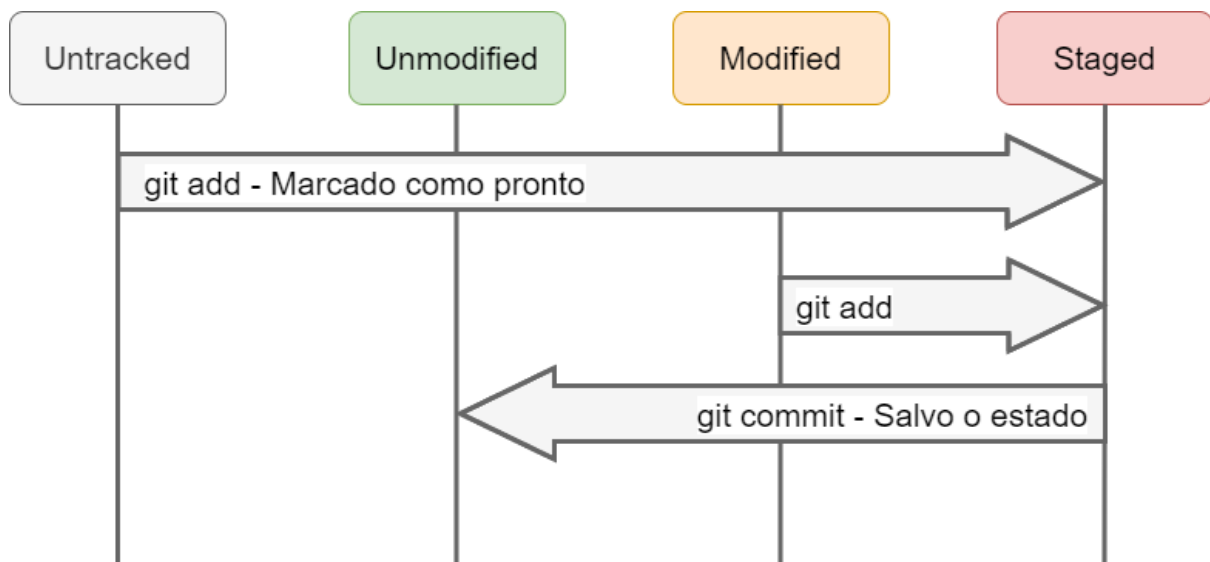
Para enviarmos arquivos para essa seção, basta commitar as alterações feitas, que estejam marcadas como prontas, ou seja, dentro da área de **Staged**. No nosso caso, temos dois arquivos: **app.py** e **projeto.ipynb**. Lembrando que um dos arquivos é um arquivo recém adicionado ao projeto, o arquivo **app.py**, enquanto o outro arquivo é um arquivo que já possuía uma versão anterior salva e que foi modificado, o arquivo **projeto.ipynb**.

Para commitar e salvar as alterações, utilizamos o comando **git commit**. É boa prática sempre fazermos um commit adicionando uma mensagem a ele, assim, quando formos verificar o histórico do repositório, saberemos o que foi feito em cada commit. E por conta disso, utilizamos o parâmetro **-m**, para podermos passar a mensagem que desejamos. No nosso caso, a mensagem será “Adicionado arquivo app.py e rodado arquivo do projeto novamente”:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git commit -m "Adicionado arquivo app.py e rodado arquivo do projeto novamente"
[main 178c9cb] Adicionado arquivo app.py e rodado arquivo do projeto novamente
2 files changed, 115 insertions(+), 115 deletions(-)
create mode 100644 app.py

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
nothing to commit, working tree clean
```

Observe que quando demos o comando, os arquivos foram enviados para a seção de **Unmodified** e agora o comando **git status** nos diz que o repositório não possui nenhuma alteração e está “limpo”.



Outro ponto importante para se notar é que, todo commit gera um código **SHA-1**, que identifica os commits. Ou seja, cada vez que fazemos um commit, o Git gera um código de identificação único para cada commit:

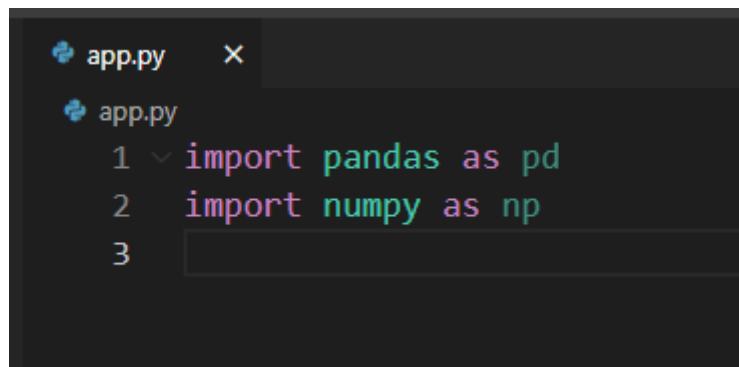
```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git commit -m "Adicionado arquivo app.py e rodado arquivo do projeto novamente"
[main 178c9cb] Adicionado arquivo app.py e rodado arquivo do projeto novamente
2 files changed, 115 insertions(+), 115 deletions(-)
create mode 100644 app.py

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
nothing to commit, working tree clean
```

E é esse código que iremos utilizar ao longo do curso para trabalharmos com o nosso repositório!

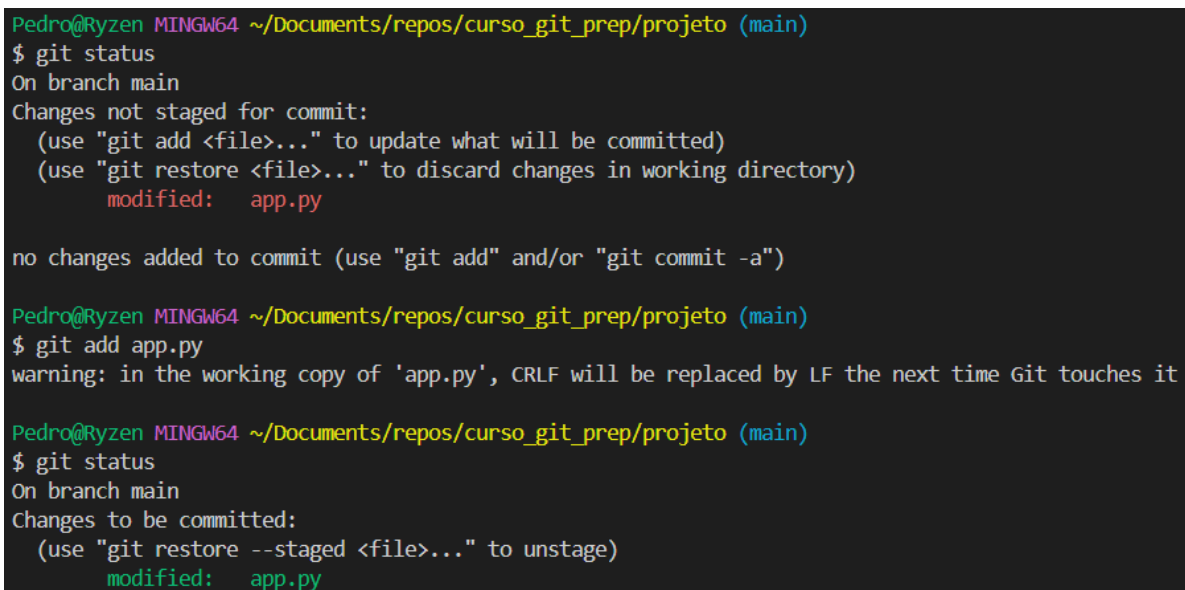
Outro ponto importante a respeito da área de **Unmodified**, Quando o arquivo entra nessa seção, ele possui basicamente dois caminhos: Ou ele é modificado e retorna ao ciclo passando pelas áreas de **Modified** → **Staged** → **Unmodified** novamente, ou ele é excluído, tendo que passar por todo o processo de e volta para a área de **Untracked**. Vamos verificar o primeiro caso.

Como não temos arquivos modificados, vamos importar as bibliotecas **Pandas** e **Numpy** em nosso arquivo **app.py**:



```
app.py x
app.py
1 import pandas as pd
2 import numpy as np
3
```

Feito isso, vamos refazer todo o processo de marcar o arquivo como pronto, utilizando o comando **git add app.py** e adicionando ele na área de **Staged**:



```
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")

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

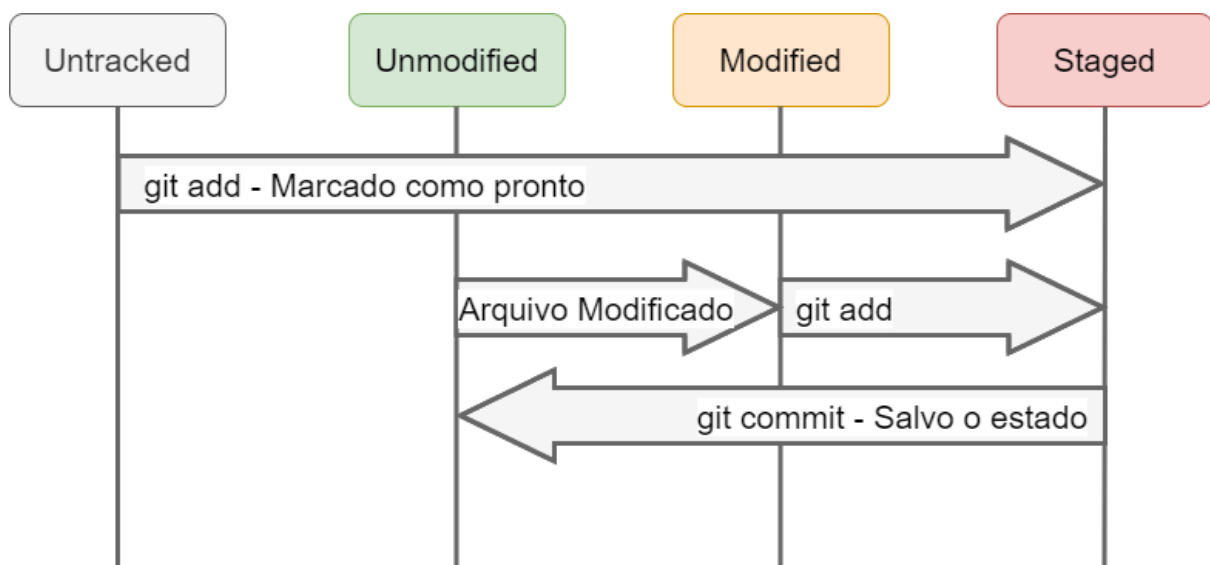
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   app.py
```

Com o arquivo marcado como pronto, vamos salvar o seu novo estado fazendo um commit e retornando ele para a seção **Unmodified** utilizando o comando **git commit** e passando a mensagem **"Adicionado bibliotecas pandas e numpy"**:

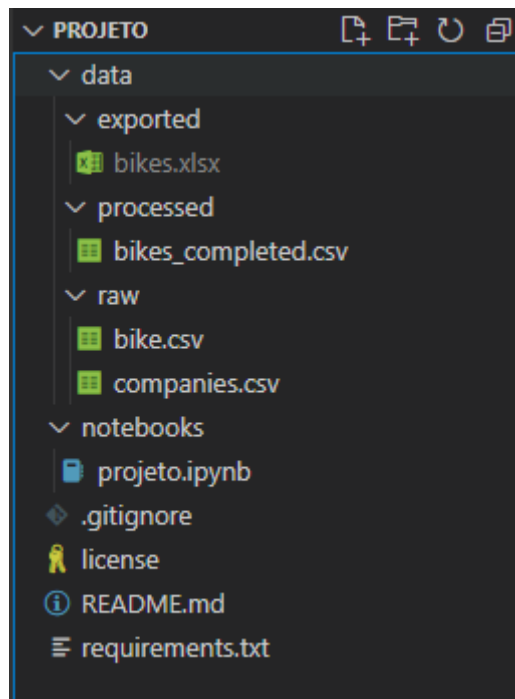
```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git commit -m "Adicionado bibliotecas pandas e numpy"
[main 35faa5d] Adicionado bibliotecas pandas e numpy
1 file changed, 2 insertions(+)

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
nothing to commit, working tree clean
```

Dessa forma, refizemos a primeira opção do ciclo de vida de um arquivo: Saiu da seção **Unmodified** para a seção **Modified** quando fizemos uma alteração no arquivo, foi adicionado na seção **Staged** a partir da seção **Modified** quando utilizamos o comando **git add** e voltou para a seção **Unmodified** quando utilizamos o comando **git commit** para salvar o novo estado do arquivo.



A outra possibilidade é a exclusão do arquivo. Para isso, vamos excluir o arquivo **app.py** do nosso projeto.



Lembrando que nosso arquivo estava dentro da seção **Unmodified**. Agora que o arquivo foi excluído, se observarmos o estado do nosso repositório Git com o comando **git status**, veremos que o Git nos informa que o arquivo foi excluído.

```
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:    app.py

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

Vamos salvar essa alteração no repositório. Ou seja, primeiro temos que adicioná-la à seção de **Staged** com o comando **git add** e depois commitar a alteração com o comando **git commit**, passando a mensagem “Removido arquivo app.py”:

```

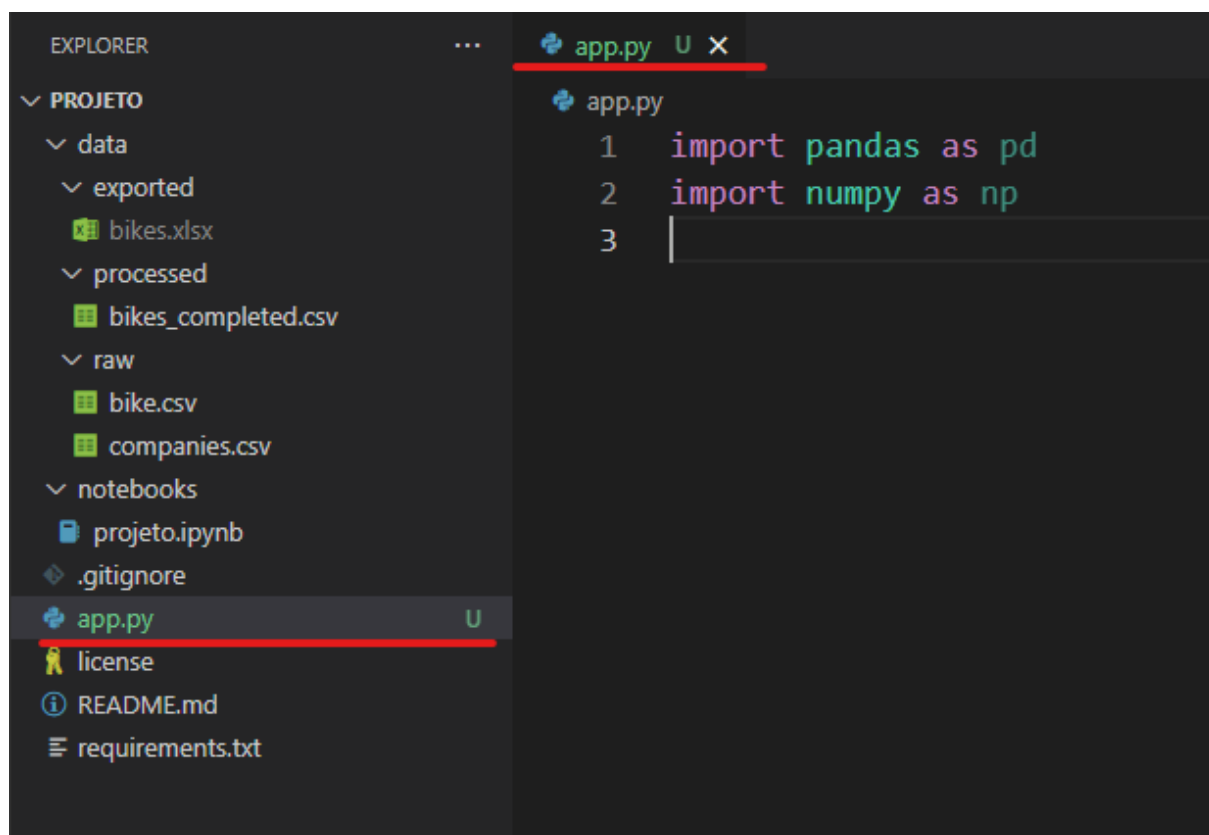
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git add app.py

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git commit -m "Removido o arquivo app.py"
[main 53f930a] Removido o arquivo app.py
 1 file changed, 2 deletions(-)
 delete mode 100644 app.py

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
nothing to commit, working tree clean

```

Com o arquivo removido e essa ação commitada, vamos criá-lo novamente para verificar que ele entrará na seção de **Untracked**, por mais que ele já tenha existido em nosso repositório:

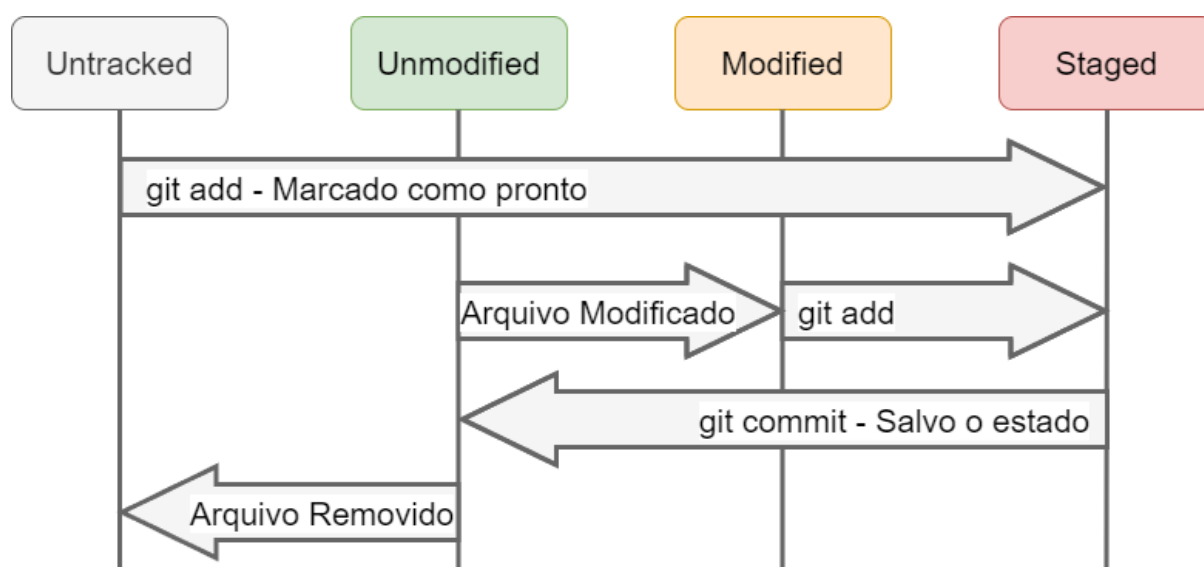


Observe que, mesmo sendo criado na mesma pasta, com o mesmo conteúdo, o arquivo está sendo entendido como um arquivo não rastreado pelo Git:

```
Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    app.py

nothing added to commit but untracked files present (use "git add" to track)
```

Ou seja, uma vez que excluimos um arquivo que esteja dentro da área de **Unmodified**, e salvamos essa alteração passando por todo o processo de commit, o arquivo deixa de ser rastreado, e se for inserido novamente ao projeto, o Git irá entendê-lo como um arquivo novo e nunca visto!



Ou seja, o ciclo de vida de um arquivo dentro de um repositório Git normalmente transita entre as seção de **Unmodified**, que é onde estão todos os arquivos rastreados, para a seção **Modified**, que é onde todos os arquivos já rastreados vão quando sofrem alguma alteração, e a seção **Staged**, que é onde os arquivos modificados vão quando são marcados como prontos com o comando **git add**, e voltam para a seção **Unmodified**, através do comando **git commit**.

Quando temos novos arquivos, ou quando eles são excluídos, utilizamos a seção **Untracked** do Git.

Para finalizar a aula, vamos commitar o arquivo que criamos nesta aula, o arquivo `app.py` com a mensagem “Arquivo app.py adicionado”:

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

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git commit -m "Arquivo app.py adicionado"
[main ea2b047] Arquivo app.py adicionado
1 file changed, 2 insertions(+)
create mode 100644 app.py

Pedro@Ryzen MINGW64 ~/Documents/repos/curso_git_prep/projeto (main)
$ git status
On branch main
nothing to commit, working tree clean
```

## Exercícios

Link para o formulário de exercícios de fixação de conteúdo: [Exercícios](#)

## Próxima Aula

Na próxima aula, iremos consolidar o que aprendemos nesta aula vendo as etapas e as áreas de versionamento utilizadas pelo Git.

## Fontes e Links Complementares

[Livro Pro Git - Capítulo 2.2 - Salvando alterações no repositório](#)