

VERSIONAMENTO DE SOFTWARE

Imagine um trabalho de escola ou de faculdade, onde escrevemos um documento para enviar aos nossos professores. Este arquivo tem versões, pois, quando escrevemos ele, começamos fazendo um pouco hoje, outro tanto amanhã, o restante depois, até chegar na versão final e entregarmos aos professores.

O mesmo acontece com código de programas: desenvolvemos o sistema em pequenos pedaços entregáveis e cada entregável é uma versão do nosso software.

É preciso entender que o versionamento não diz respeito apenas a uma classificação numérica. Trata-se de uma ferramenta de controle, uma metodologia que permite gerir atualizações ou correções e planejar intervenções futuras, tornando o trabalho do desenvolvedor cada vez mais prático, eficiente e fidedigno.

Benefícios do versionamento de software

- **Análise de ameaças:** Em uma realidade marcada pela interconectividade e por organizações globais, as ameaças à segurança dos sistemas responsáveis por essas conexões têm o potencial de gerar grandes prejuízos. O processo de versionamento permite a proteção do código-fonte do sistema. Já o sequenciamento lógico das atualizações serve como registro histórico para a análise dos erros e ataques sofridos, o que possibilita também a reformulação dos sistemas para versões mais robustas.
- **Reparação de bugs:** O controle de versão fornece ao desenvolvedor o acesso total à codificação realizada em determinado programa ao longo do tempo. Essa facilidade favorece um alto grau de qualidade do trabalho realizado pelos programadores, uma vez que todo o trabalho pode ser revisto, alterado e corrigido.
- **Alterações de arquitetura:** Devido à velocidade com que a tecnologia evolui, muitas alterações precisam ser implementadas nas mais diferentes etapas de desenvolvimento — principalmente após a entrega do software. Graças ao versionamento, é possível acompanhar tal evolução de maneira muito mais fácil e prática no que diz respeito tanto aos controles internos dos profissionais desenvolvedores quanto ao reconhecimento das alterações por parte do usuário final.
- **Potencialização do trabalho colaborativo:** Por meio de um processo de versionamento, é possível reconhecer facilmente qualquer alteração nos programas utilizados, tornando o trabalho em conjunto mais produtivo e sem riscos à segurança da informação.
- **Upgrades estéticos:** Buscando um aprimoramento, melhorias são adotadas com layouts responsivos, ao passo que designs mais atrativos, cores, botões e imagens chamativas provocam alterações significativas de códigos. Sem um sistema de controle de versão, a atividade de acompanhamento em cada uma dessas alterações se torna muito complicada.

Git e GitHub

Git:

- **Git é um sistema de controle de versionamento de software.** Ele é baseado em repositório, que contém todas as versões do código e as cópias de cada desenvolvedor.

- O Git é um projeto de código aberto maduro e com manutenção ativa desenvolvido em 2005 por Linus Torvalds, o famoso criador do kernel do sistema operacional Linux. É um exemplo de DVCS (portanto, Sistema de Controle de Versão Distribuído). Em vez de ter apenas um único local para o histórico completo da versão do software, como é comum em sistemas de controle de versão outrora populares como CVS ou Subversion (também conhecido como SVN), no Git, a cópia de trabalho de todo desenvolvedor do código também é um repositório que pode conter o histórico completo de todas as alterações.
- Todos os objetos do Git são protegidos com criptografia, para evitar alterações indevidas e maliciosas.

GitHub

- **O GitHub é uma rede social de desenvolvedores.** A primeira parte do nome, “Git”, é por causa da utilização do sistema de controle de versão e a segunda parte, “Hub”, tem a ver com a conexão entre profissionais de programação de qualquer lugar do mundo.



Dê ao seu código uma casa na nuvem

Grave ou rebobine qualquer alteração no seu código para manter você e sua equipe em sincronia. **Hospede tudo gratuitamente com repositórios públicos e privados ilimitados.**

Github.com

<https://www.ipsense.com.br/blog/entenda-a-importancia-do-versionamento-de-software/>

<https://woliveiras.com.br/posts/introdu%C3%A7%C3%A3o-a-versionamento-de-c%C3%B3digo-e-conhecendo-o-git/>

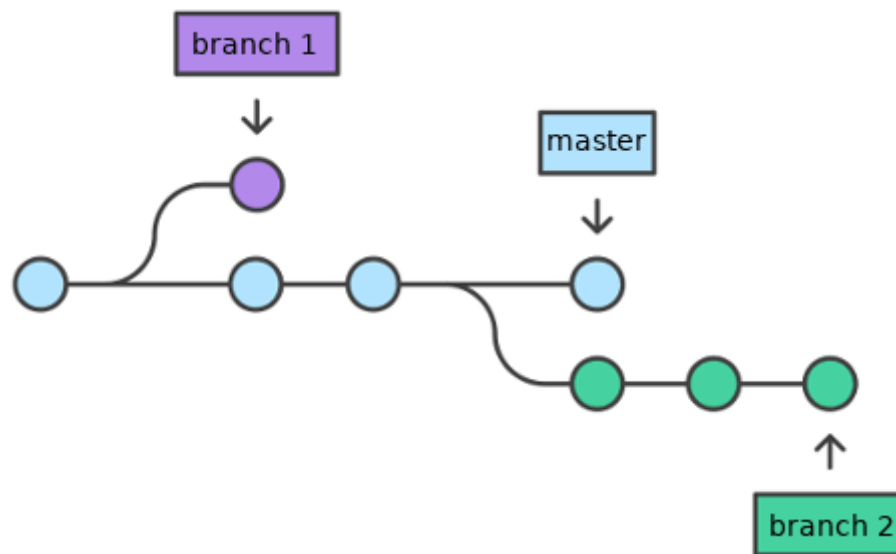
Repositório

Repositório, ou repo, é um diretório onde os arquivos do seu projeto ficam armazenados. Ele pode ficar em um depósito do GitHub ou em seu computador. Você pode armazenar códigos, imagens, áudios, ou qualquer outra coisa relacionada ao projeto no diretório.

Branch

Branch é uma cópia do diretório. Você pode usar o branch para desenvolver isoladamente.

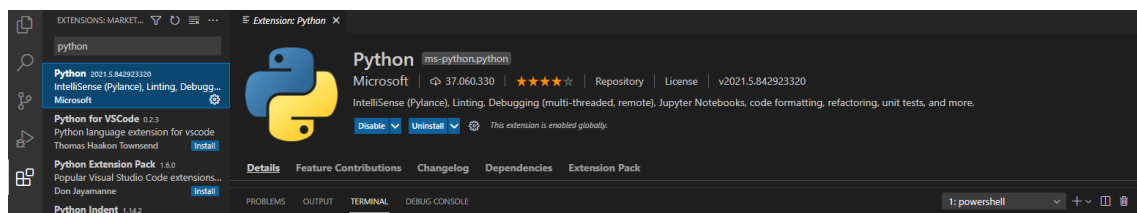
Trabalhar em um branch não irá afetar o repositório central ou outros branches. Depois de finalizar o trabalho você pode combinar seu branch isolado com outros branches



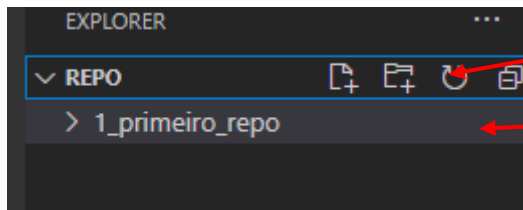
Instalando o Git: Acessar <https://git-scm.com/downloads>

Instalando o VS Code: <https://code.visualstudio.com/>

Interessante instalar o Python no Vs Code, pois ele pode auxiliar na codificação do Git.



Criando repositório



Pasta onde vou salvar meus repositórios

Pasta para cada projeto

Entrar na pasta do projeto para criar repositório

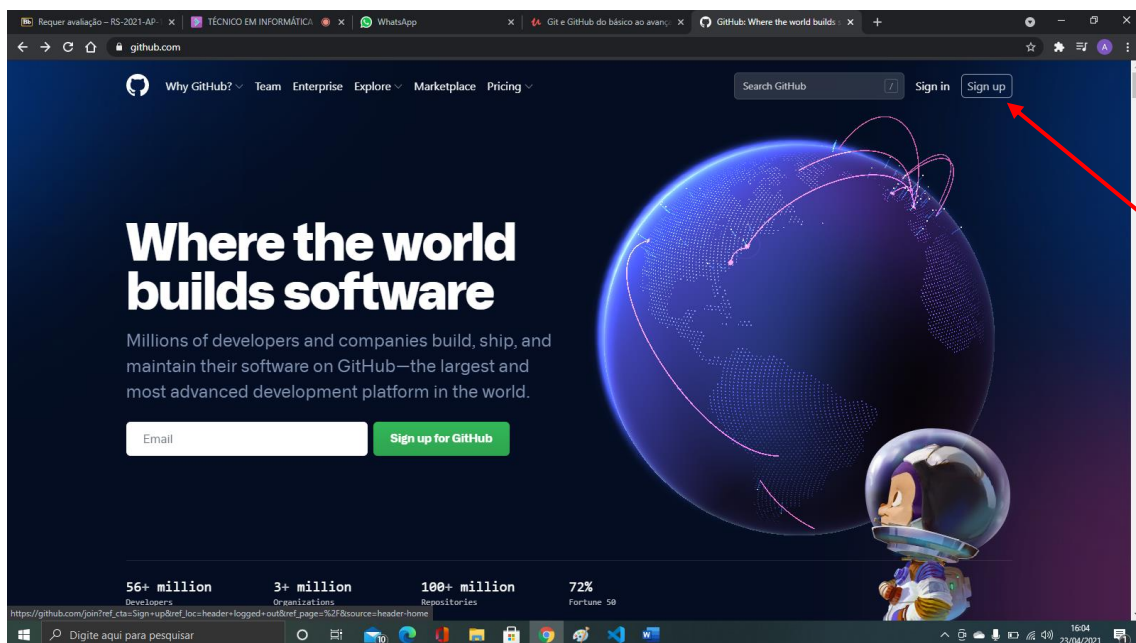
No terminal do Vs Code, digitar:

```
PS D:\TDS\UC10 - versionando codigos\repo> cd .\1_primeiro_repo\
```

git status para verificar se algum repositório já foi criado.

Caso nenhum repositório tenha sido criado, criar com **git init**

Criando conta no GitHub



Join GitHub

Create your account

Username *

Email address *

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.
[Learn more.](#)

Email preferences

☐ Send me occasional product updates, announcements, and offers.

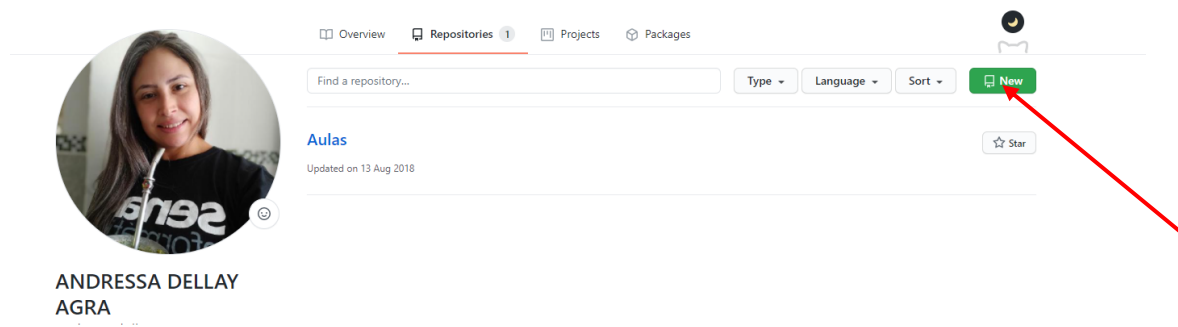
Verify your account

Resolva este enigma para sabermos que
você é uma pessoa de verdade

Verificar

Enviando repositório para github

Criar um novo repositório:




The screenshot shows the GitHub profile of ANDRESSA DELLAY AGRA. The profile includes a circular profile picture, the name ANDRESSA DELLAY AGRA, and a bio. The repository list is visible, showing a repository named 'Aulas' updated on 13 Aug 2018. A red arrow points to the 'New' button in the repository list.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 andressadellay ▾

Repository name *

/ versionamentodesoftware ✓

Great repository names are short and memorable. Need inspiration? How about [miniature-garbanzo?](#)

Description (optional)

Repositório para as aulas de versionamento de software

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

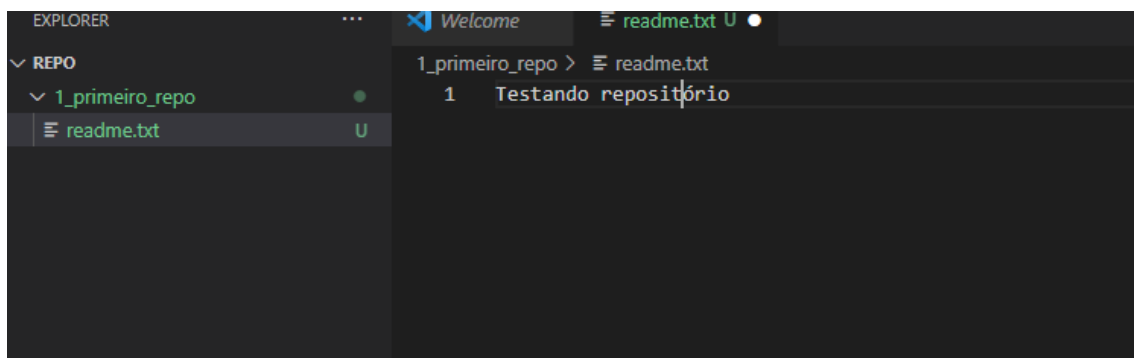
Arquivo que vai explicar o que tem no repositório. Descrição do projeto em si

Arquivo que se adiciona para ignorar outros arquivos que o git monitore

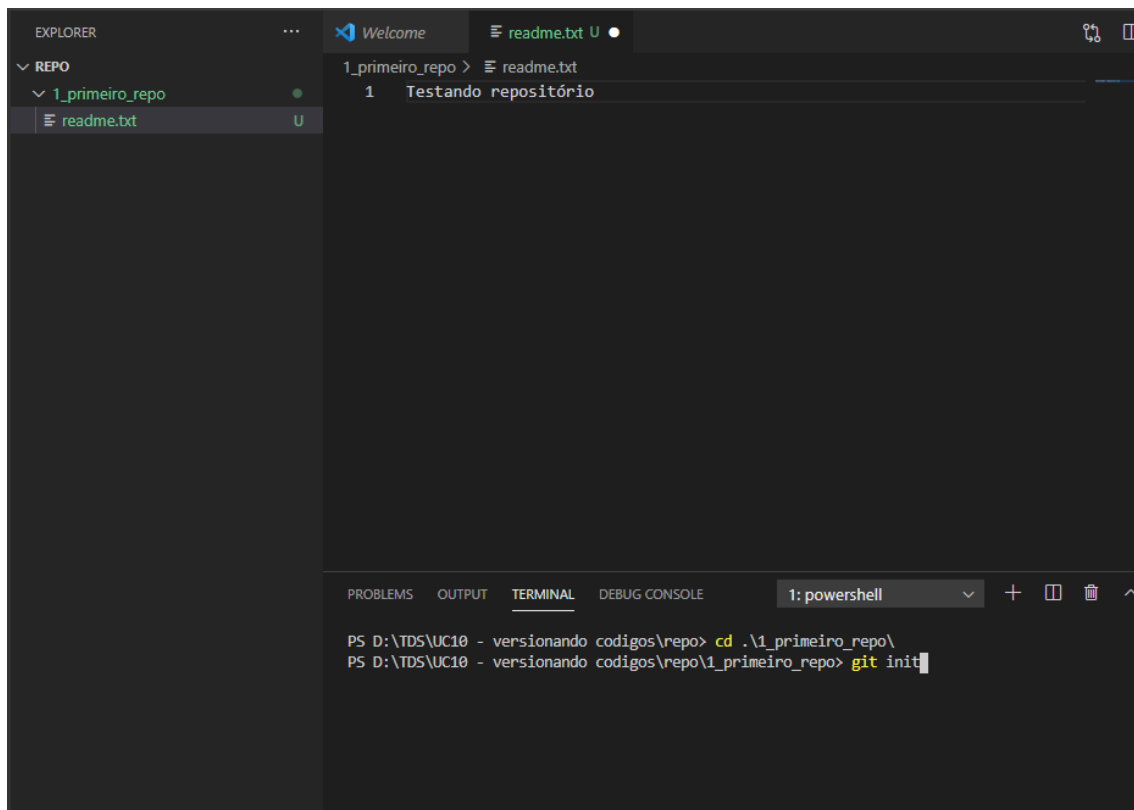
Quando se tem um software e quer atribuir uma licença a ele. Diz o que os outros podem fazer ou não com o seu código.

```
echo "# versionamentodesoftware" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/andressadellay/versionamentodesoftware.git
git push -u origin main
```

Criar um arquivo com extensão txt



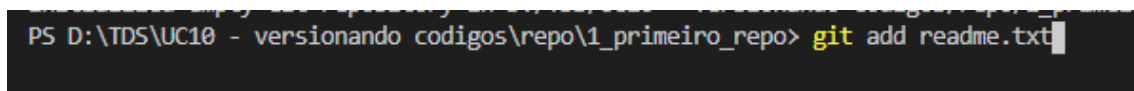
Inicializar o repositório



The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows a folder named '1_primeiro_repo' containing a file 'readme.txt'. The main editor area shows the 'readme.txt' file with the text '1 Testando repositório'. At the bottom, the Terminal pane shows the following commands and output:

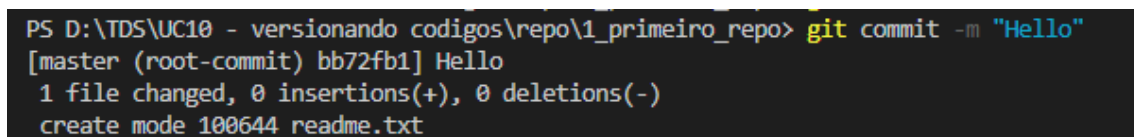
```
PS D:\TDS\UC10 - versionando codigos\repo> cd .\1_primeiro_repo\  
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git init
```

Adicionar o redme.txt ao git



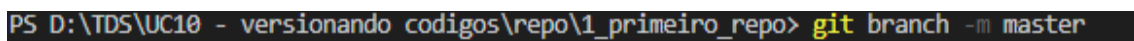
```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git add readme.txt
```

Enviando



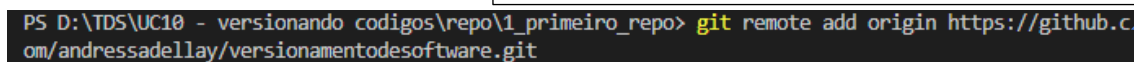
```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git commit -m "Hello"  
[master (root-commit) bb72fb1] Hello  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 readme.txt
```

Criando uma branch- uma ramificação do meu projeto que é master (sempre terá essa banch em todos os projetos)



```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git branch -m master
```

Sincronizando



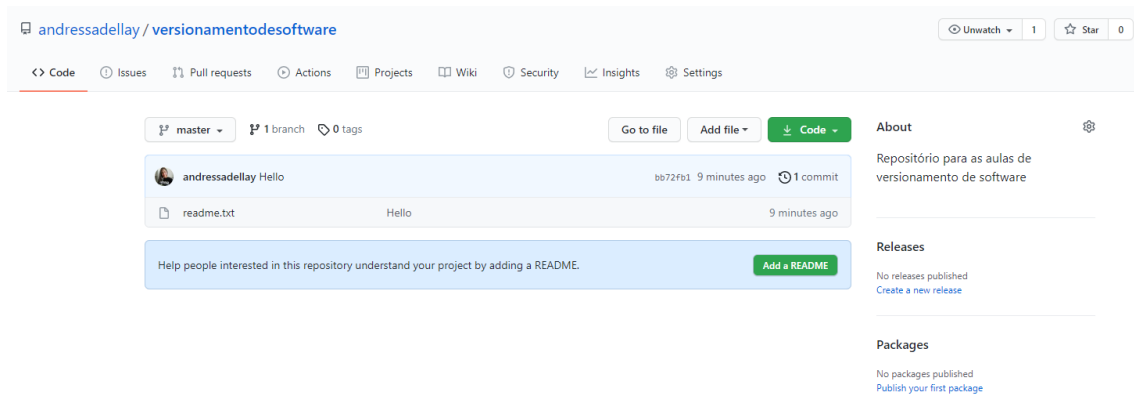
```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git remote add origin https://github.com/andressadellay/versionamentodesoftware.git
```

Caso não der certo o link, ir no GitHub e escolher a opção HTTPS. Para colar, botão direito do mouse ou ctrl+shift+v

Enviando para a master da origem – pode pedir para permitir acesso no Git Hub instalado na máquina.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git push -u origin master
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 221 bytes | 221.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/andressadellay/versionamentodesoftware.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

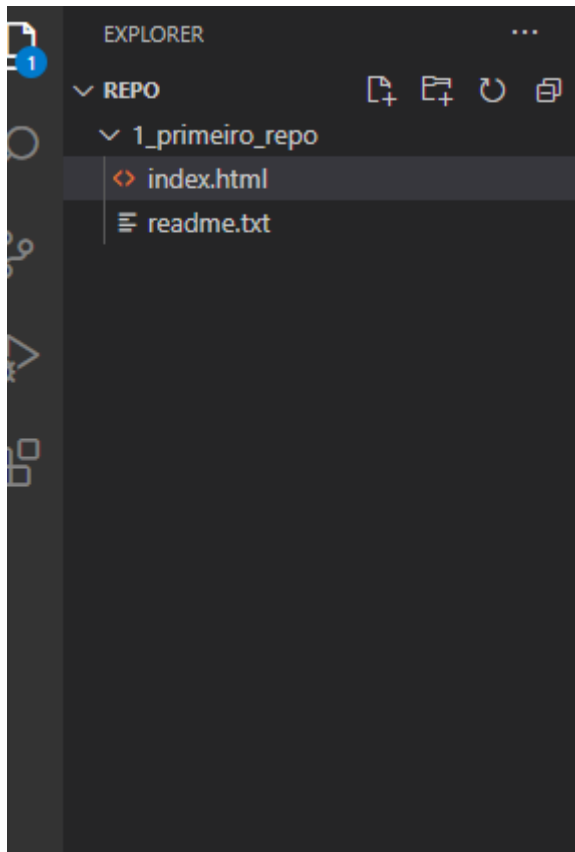
Arquivo enviado



VERIFICANDO MUDANÇAS NO PROJETO

As mudanças do projeto podem ser verificadas por **git status**

Criar um novo arquivo, neste exemplo index.html



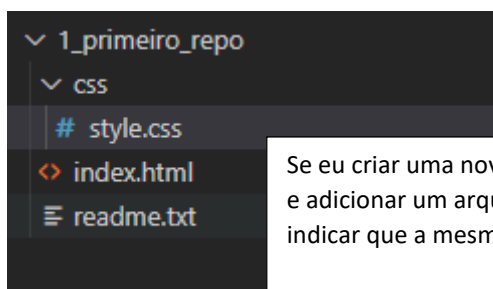
```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git status
On branch master
Your branch is up to date with 'origin/master'.

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

nothing added to commit but untracked files present (use "git add" to track)
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo>
```

Arquivo ainda não mapeado pelo git

Sugere dar git add para verificar quais mudanças estão sendo adicionados ao arquivo.



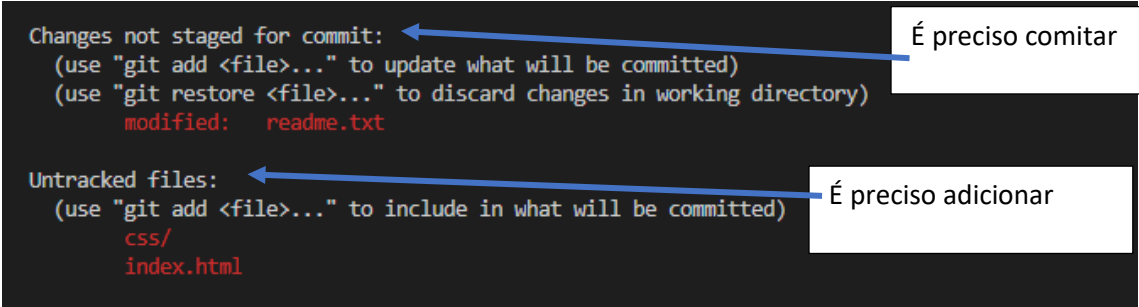
Se eu criar uma nova pasta, como por exemplo a pasta css e adicionar um arquivo dentro e verificar o status, vai me indicar que a mesma não foi mapeada pelo git ainda

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git status
On branch master
Your branch is up to date with 'origin/master'.

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

nothing added to commit but untracked files present (use "git add" to track)
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo>
```

Caso se modifique um arquivo que tenha sido comitado anteriormente e se verificar o status, o mesmo retornará Changes not staged.



```
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:   readme.txt

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

Annotations:

- ← É preciso comitar (points to "Changes not staged for commit")
- ← É preciso adicionar (points to "Untracked files")

ADICIONANDO ARQUIVOS NOVOS AO GIT

Para adicionarmos arquivos novos a um projeto, utilizamos o comando **git add**

Podemos adicionar um arquivo específico ou vários de uma vez.

Somente adicionando arquivos, eles serão monitorados pelo git, o seja, se não adicionar o arquivo, ele não estará no controlo de versão

Adicionando arquivo:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git add index.html
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> 
```

Após ter adicionado o arquivo, ao verificar status, o mesmo indica que ao ser comitado o projeto, o arquivo também será.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   index.html

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:   readme.txt

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

PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> 
```

Adicionado mais de um arquivo/pasta ao mesmo tempo:

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git add .
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   css/style.css
    new file:   index.html
    modified:   readme.txt
```

SALVANDO ALTERAÇÕES DO PROJETO

As alterações salvas do projeto são realizadas por **git commit**

Podemos comitar arquivos específicos ou vários, utilizando a flag **-a** (de all)

É uma boa prática enviar uma mensagem a cada commit, informando as alterações de foram feita. Utiliza-se a flag **-m** (de message)

Comitando apenas um arquivo:

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git commit readme.txt -m "Testando commit de um arquivo específico"
[master 407ddde] Testando commit de um arquivo específico
1 file changed, 1 insertion(+)
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> █
```

Commit de todos os arquivos de uma única vez:

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git commit -a -m "Commit de todos os arquivos do projeto"
[master a2475a4] Commit de todos os arquivos do projeto
2 files changed, 3 insertions(+)
create mode 100644 css/style.css
create mode 100644 index.html
```

ENVIANDO CÓDIGO AO REPO REMOTO

Quando finalizamos uma funcionalidade nova, enviamos o código ao repositório remoto, que é o código-fonte.

Esta ação é feita pelo **git push**

Após esta ação, o código do servidor será atualizado baseando-se no código local enviado






Estado ideal para fazer o push:

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Enviando projeto ao repositório:

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (7/7), 682 bytes | 170.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/andressadellay/versionamentodesoftware.git
bb72fb1..a2475a4 master -> master
```

	andressadellay Commit de todos os arquivos do projeto	a2475a4 15 minutes ago	 3 commits
	css	Commit de todos os arquivos do projeto	15 minutes ago
	index.html	Commit de todos os arquivos do projeto	15 minutes ago
	readme.txt	Testando commit de um arquivo específico	16 minutes ago

RECEBENDO AS MUDANÇAS

Muitas vezes é necessário sincronizar o projeto local com as mudanças do remoto;

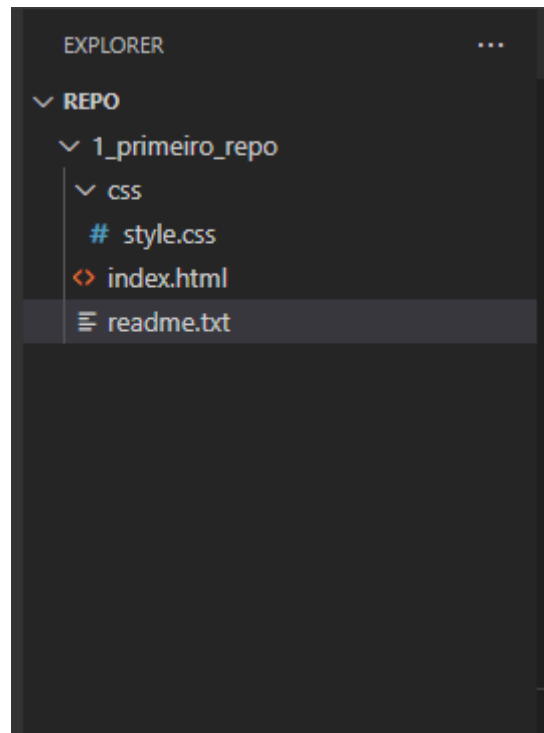
Esta ação é feita pelo git pull;

Após o comando, são realizadas buscas das atualizações, se encontradas, serão unidas ao código atual existente na máquina local.

Criando um arquivo novo no git para realizar o teste:

The screenshot displays the GitHub web interface for a repository named 'versionamentodesoftware'. At the top, it shows the 'master' branch with 1 branch and 0 tags. A dropdown menu for 'Add file' is open, showing options to 'Create new file' or 'Upload files'. Below this, a list of files is shown: 'css', 'index.html', and 'readme.txt'. The 'readme.txt' file is selected, and its content is displayed: 'Testando repositório Andressa'. Below the file content, there is a 'Commit new file' dialog box. The dialog box has a title 'Commit new file' and a subtitle 'Create about.html'. It contains a text area for 'Add an optional extended description...'. At the bottom, there are two radio buttons: 'Commit directly to the master branch.' (selected) and 'Create a new branch for this commit and start a pull request. Learn more about pull requests.' Below the radio buttons are two buttons: 'Commit new file' and 'Cancel'.

Antes do comando git pull, o arquivo dentro do repositório não encontra-se localmente na máquina:

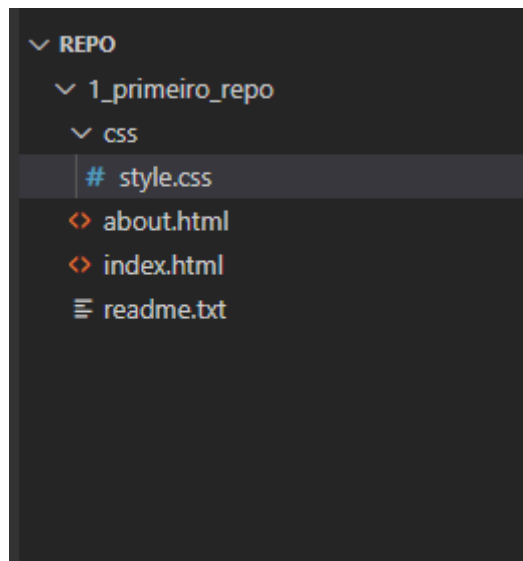


Comando **git pull**

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), 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), 776 bytes | 4.00 KiB/s, done.
From https://github.com/andressadellay/versionamentodesoftware
   a2475a4..9cdac86  master    -> origin/master
Updating a2475a4..9cdac86
Fast-forward
 about.html | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 about.html
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> 
```

Após o comando, todos os arquivos existentes dentro do git, aparecem na máquina local:



O **comando git pull** não vale apenas para arquivos novos, mas também arquivos que já existem na máquina local e sofrem alteração no git.

CLONANDO REPOSITÓRIO

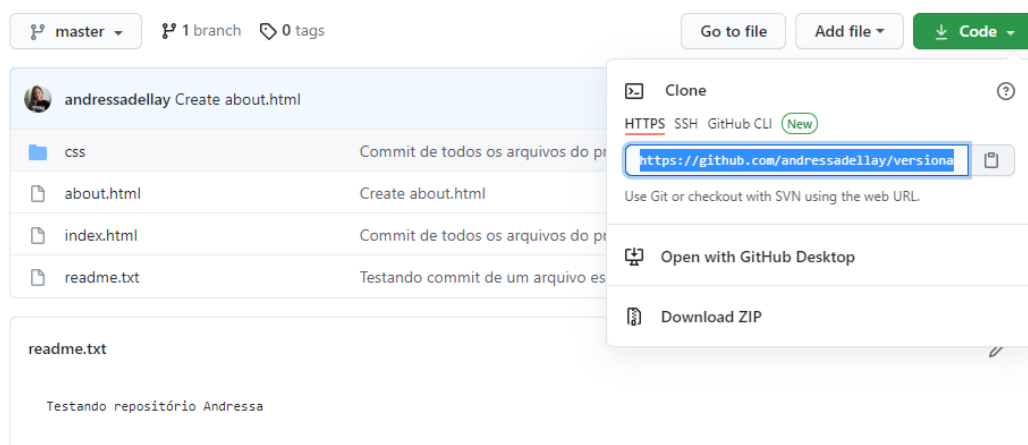
O ato de baixar um repositório de um servidor remoto é chamado de clonar repositório;

Para esta ação utilizamos **git clone**;

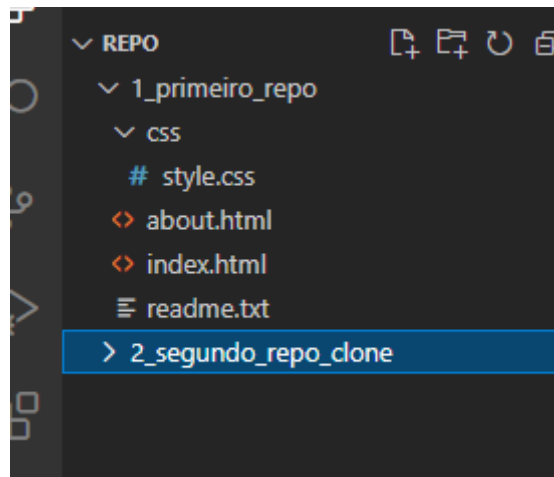
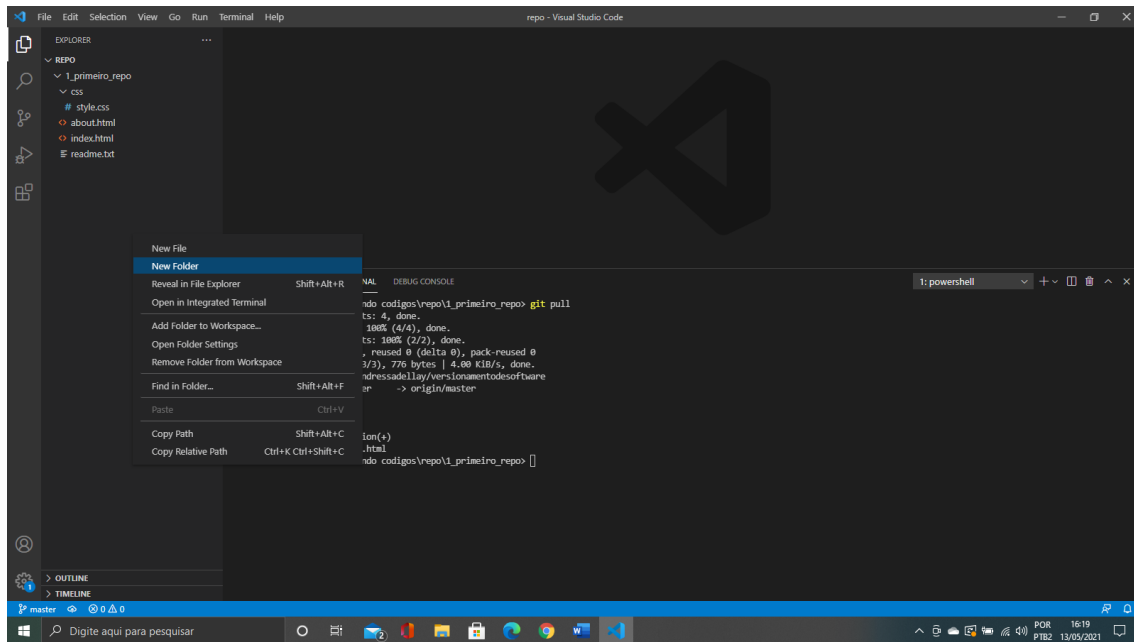
Passando a referência do repositório remoto;

Este comando é utilizado quando entramos em um novo projeto, por exemplo.

Copiar o link do code:



Criando uma nova pasta para testar clone de repositório:



Comando git clone com o link copiado:



O ponto significa para clonar o link para o diretório atual.

Clonagem realizada:

```
2_segundo_repo_clone
├── css
├── < about.html
├── < index.html
└── ≡ readme.txt
```

REMOVENDO ARQUIVOS DO REPOSITÓRIO

O comando para deletar arquivos de monitoramento do git é **git rm**.

Após deletar o arquivo do git, ele não terá mais suas atualizações consideradas pelo git, apenas se for adicionado novamente, pelo comando **git add**.

```
PS D:\TDS\UC10 - versionando codigos\repo> cd .\1_primeiro_repo\
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git rm readme.txt
rm 'readme.txt'
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    readme.txt

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:   index.html
```

Comando para excluir documento

Verifica status do git

Indica que é necessário fazer um commit

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git commit -a -m "Deletando arquivo desnecessário"
[master 60df33b] Deletando arquivo desnecessário
 2 files changed, 12 insertions(+), 1 deletion(-)
 delete mode 100644 readme.txt
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 457 bytes | 228.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/andressadellay/versionamentodesoftware.git
 9cdac86..60df33b master -> master
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo>
```

Realizando a atualização dos arquivos no git. Lembrando: -a (atualiza todos os arquivos) -m (mensagem)

Envia atualização ao repositório

HISTÓRICO DE ALTERAÇÕES

É possível verificar o histórico de commits que ocorreram dentro do git, para isso, utilizamos o comando **git log**.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git log
commit 60df33baafb07b568af25b5534c1a6be99348f82 (HEAD -> master, origin/master)
Author: Andressa Dellay Agra <andressadellayagra@gmail.com>
Date:   Wed Jun 2 18:57:51 2021 -0300

    Deletando arquivo desnecessário

commit 9cdac86ffb78c2ba6a63565608653bde2e4009ff
Author: ANDRESSA DELLAY AGRA <42360340+andressadellay@users.noreply.github.com>
Date:   Thu May 13 15:54:41 2021 -0300

    Create about.html

commit a2475a4ba23b67536481e657bc0597d960301ee5
Author: Andressa Dellay Agra <andressadellayagra@gmail.com>
Date:   Fri May 7 16:10:16 2021 -0300

    Commit de todos os arquivos do projeto

commit 407dde12b24e51e5facc87c1ea4e78b614d0c25
Author: Andressa Dellay Agra <andressadellayagra@gmail.com>
Date:   Fri May 7 16:09:07 2021 -0300

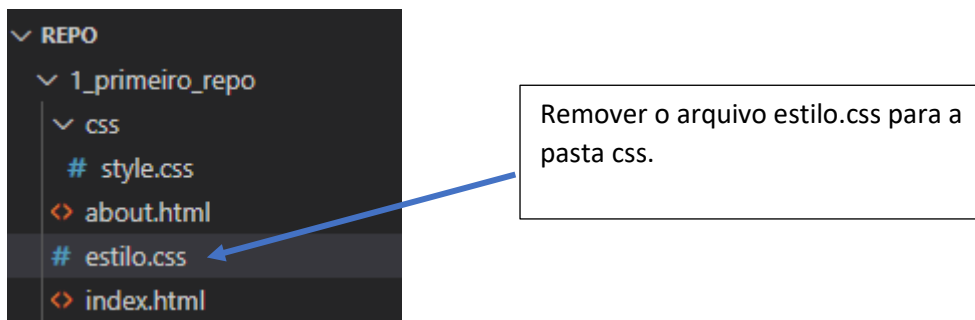
    Testando commit de um arquivo específico
```

Para continuar verificando todas as ações, é necessário dar enter.

Caso queira **encerrar** o processo de **log** no meio, basta dar **ctrl+c** ou em alguns casos apenas **Q**.

RENOMEANDO NOMES DE ARQUIVOS OU TROCANDO DE PASTAS

Para renomear um arquivo ou remover ele de uma pasta e posicionar em outra, utilizamos o comando **git mv**.



```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git add .
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   estilo.css

PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git commit -a -m "Enviando arquivo folha de estilo"
[master 612a613] Enviando arquivo folha de estilo
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 estilo.css
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 362 bytes | 181.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/andressadellay/versionamentodesoftware.git
 60df33b..612a613 master -> master
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git pull
Already up to date.
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo>
```

Adicionando arquivo estilo.css ao git

Verificando status do git, que responde ter um novo arquivo denominado "estilo.css"

Realizando commit de todos os possíveis novos arquivos.

Envia atualização ao repositório

Busca atualizações do repositório

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git mv estilo.css .\css\
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  renamed:   estilo.css -> css/estilo.css
```

Movendo o arquivo para a pasta css

Verificando status do git

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git commit -a -m "Movendo o arquivo estilo.css para a pasta css"
[master 744edfe] Movendo o arquivo estilo.css para a pasta css
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename estilo.css => css/estilo.css (100%)
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 410 bytes | 136.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/andressadellay/versionamentodesoftware.git
 612a613..744edfe master -> master
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git pull
Already up to date.
```

Realizando o commit

Envia atualização ao repositório

Busca atualizações do repositório

```
▼ REPO
  ▼ 1_primeiro_repo
    ▼ css
      # estilo.css
      # style.css
      <> about.html
      <> index.html
```

Arquivo movido para a pasta indicada

Renomear arquivo style.css para folha_de_estilo.css

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git mv .\css\style.css .\css\folha_de_estilo.css
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:   css/style.css -> css/folha_de_estilo.css

PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git commit -a -m "Renomeando arquivo"
[master 85f37a5] Renomeando arquivo
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename css/{style.css => folha_de_estilo.css} (100%)
```

Renomeando o arquivo style.css para
folha_de_estilo.css

```
▼ REPO
  ▼ 1_primeiro_repo
    ▼ css
      # estilo.css
      # folha_de_estilo.css
      <> about.ntml
      <> index.html
```

DESAFIZENDO ALTERAÇÕES

O arquivo original pode ser retomado ao seu estado original, ou seja, como ele se encontra no repositório (git hub).

O comando utilizado para desfazer alterações é o **git checkout**.

Importante ressaltar que este comando deve ser utilizado caso sejam feitas inúmeras alterações em um determinado arquivo e o mesmo ainda não tenha sido enviado para o repositório.

```
# folha_de_estilo.css
1_1_primeiro_repo > css > # folha_de_estilo.css > h1
1  h1{
2    color: blue;
3  }
4  p{
5    font-size: 20px;
6  }
7  h1{
8    color: pink;
9  }
```

Por exemplo: inseri os novos comandos da linha 4 até 9, porém quero voltar ao estado original, ou seja, ao conteúdo que está no git hub.

```
nothing to commit, working tree clean
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git status
On branch master
Your branch is up to date with 'origin/master'.

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:   css/folha_de_estilo.css

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo>
```

Verifico através do git status que há modificações em um arquivo na máquina local.

Solicito que todas as alterações locais sejam desfeitas e que o arquivo retorne ao estado original do repositório. Automaticamente, minha máquina local recebe o estado original do arquivo.

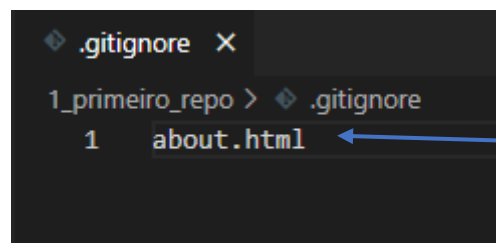
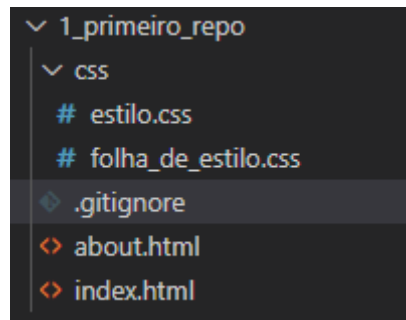
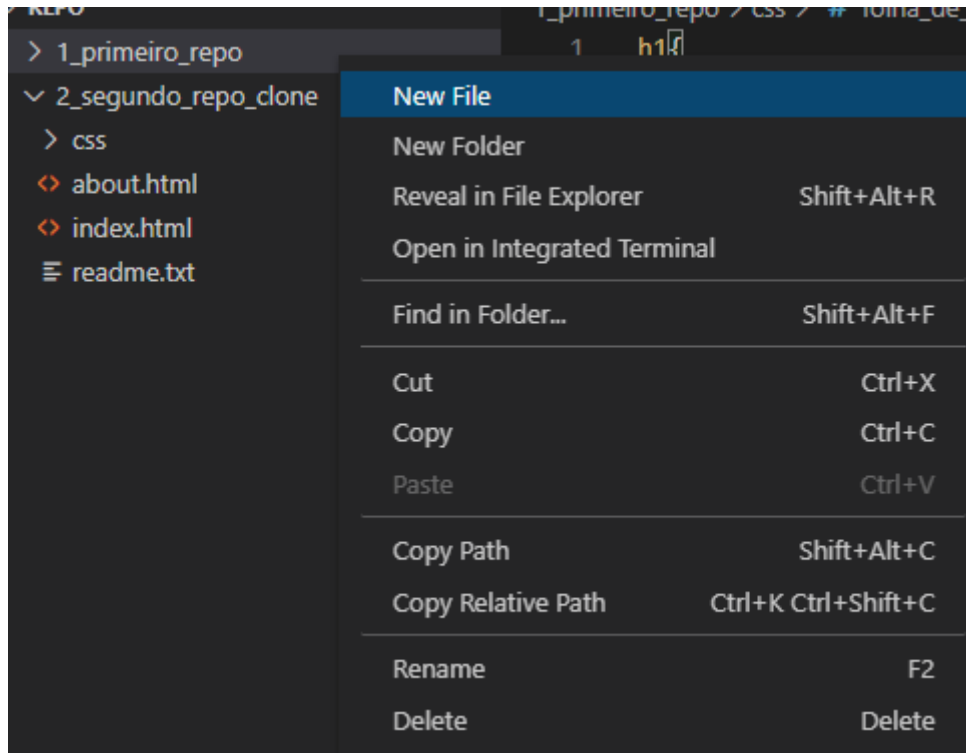
```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git checkout .\css\folha_de_estilo.css
Updated 1 path from the index
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo>
```

```
# folha_de_estilo.css X
1_1_primeiro_repo > css > # folha_de_estilo.css > h1
1  h1{
2    color: blue;
3  }
```

IGNORANDO ARQUIVOS NO PROJETO – para que não sejam enviados ao GitHub

Para **ignorar arquivos ou pastas** ao enviar atualizações ao git hub, devemos **criar um arquivo** denominado **.gitignore** na raiz do projeto em que estamos trabalhando.

Útil para arquivos que são gerados automaticamente ou que possuam dados sensíveis.



Ignorando o arquivo
about.html

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git status
On branch master
Your branch is up to date with 'origin/master'.
```

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

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git add .
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git commit -a -m "Enviando .gitignore"
[master afbf5de] Enviando .gitignore
1 file changed, 1 insertion(+)
create mode 100644 .gitignore
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 366 bytes | 183.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/andressadellay/versionamentodesoftware.git
85f37a5..afbf5de master -> master
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git pull
Already up to date.
```

```
▼ REPO
  ▼ 1_primeiro_repo
    ▼ css
      # estilo.css
      # folha_de_estilo.css
      ◆ .gitignore
      <> about.html
      <> index.html
```

A cor do arquivo about.html muda, pois não será enviado para o git hub

```
◆ .gitignore X
1_primeiro_repo > ◆ .gitignore
1  about.html
2  css/*
```

Ignorando a pasta e tudo que estiver dentro dela.

```
▼ REPO
  ▼ 1_primeiro_repo
    ▼ css
      # estilo.css
      # folha_de_estilo.css
      ◆ .gitignore
      <> about.html
      <> index.html
```

Ignorando a pasta e tudo que estiver dentro dela.

DESAFANDO TODAS AS ALTERAÇÕES

Podemos resetar as mudanças feitas no projeto, para o estado do último push feito.

Todas as alterações comitadas e também pendentes serão excluídas.

O comando utilizado é o git reset e normalmente se utiliza a flag `--hard`.

Exemplo: Alterando os arquivos `index.html` e `folha_de_estilo.css`:

```
index.html • # folha_de_estilo.css •
1_primeiro_repo > < index.html > html > body > p
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6  |   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7  |   <title>Document</title>
8  </head>
9  <body>
10 |   <p> Olá mundo</p>
11 </body>
12 </html>
```

```
1_primeiro_repo > css > # folha_de_estilo.css > p
1  h1{
2  |   color: blue;
3  }
4  p{
5  |   color: blueviolet;
6  }
```

```
PS D:\TDS\UC10 - versionando codigos\repo> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  1_primeiro_repo/
  2_segundo_repo_clone/

nothing added to commit but untracked files present (use "git add" to track)
PS D:\TDS\UC10 - versionando codigos\repo>
```

Verificando o status do projeto, onde retorna que 2 arquivos não foram comitados.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git commit -a -m "Comitando mudanças"
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
PS D:\TDS\UC10 - versionando codigos\repo\1_primeiro_repo> git status
On branch master
Your branch is up to date with 'origin/master'.
```

Reseta todos os arquivos do projeto para o último push realizado.

Obs: Os arquivos locais deve ser sito salvos e depois comitados para que sejam resetados.

BRANCH

Quando um projeto é criado, ele inicia na branch master (estamos trabalhando nela até aqui).

Branch é a forma que o git separa as versões dos projetos.

Normalmente cada nova feature de um projeto fica em um branch separada, somente após estar com a feature completa é que passamos o conteúdo do branch para o branch master.

Após a finalização das alterações os branches são unidos para ter o código-fonte final.

O branch principal do projeto é sempre o MASTER.

CRIANDO E VISUALIZANDO OD BRANCHES

Para visualizar os branches disponíveis o comando utilizado é o **git branch**.

Para criar um branch o comando utilizado é o **git branch <nome_da_branch>**.

```
✓ REPO
> 1_branches
> 1_primeiro_repo
> 2_segundo_repo_clone
```

Crie uma nova pasta de projeto para poder exercitar as branches.

```
PS D:\TDS\UC10 - versionando codigos\repo> cd .\1_branches\
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git clone https://github.com/andressadellay/versionamentodesoftware.git .
Cloning into '.'...
remote: Enumerating objects: 30, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 30 (delta 3), reused 26 (delta 1), pack-reused 0
Receiving objects: 100% (30/30), done.
Resolving deltas: 100% (3/3), done.
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> |
```

Para poder exercitar, vou clonar um dos projetos para dentro de 1_branches.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git branch
* master
```

Verificar as branches existentes, que neste caso é a master. O * significa que estamos nela.

Criando uma nova branch.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git branch primeiro_branch
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git branch
* master
  primeiro_branch
```

DELETANDO BRANCHES

Podemos deletar um branch com a flag **-d** ou **--delete**.

Geralmente se usa o delete quando o branch foi criado errado.

Não é comum deletar um branch, normalmente guardamos o histórico do trabalho.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git branch teste
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git branch
* master
  primeiro_branch
  teste
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git branch -d teste
Deleted branch teste (was 4ffb25a).
```

-d ou --delete poderia ser utilizado.

MUDANDO DE BRANCH

Podemos mudar para outro branch utilizando o comando **git checkout -b <nome_da_branch>**.

O comando apresentado também é utilizado para dispensar mudanças de um arquivo.

Alterando o branch podemos levar alterações que não foram comitadas junto.

IMPORTANTE:

- a)** Sempre devemos commitar atualizações antes da troca de branches. Caso contrário ao trocar de branch, as atualizações irão junto.
- b)** Sempre que for criar um novo branch, criar a partir da master. Por exemplo, se estiver na branch teste_criacao, sair desse branch, ir para a master e a partir daí criar outro branch.
- c)** Nunca esquecer de dar um git pull na nova branch, para poder receber todas as atualizações da master.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git branch
* master
  primeiro_branch
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git checkout primeiro_branch
Switched to branch 'primeiro_branch'
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git branch
  master
* primeiro_branch
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> 
```

Mudando para branch já existente.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git checkout -b teste_criacao
Switched to a new branch 'teste_criacao'
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git branch
  master
  primeiro_branch
* teste_criacao
```

Criando e mudando para uma nova branch.

```
<> index.html ●
1_branches > <> index.html > html > body > p
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
10     <p>Trabalhando com branches</p>
11 </body>
12 </html>
```

Modifico um arquivo dentro da branch selecionada "teste_criacao" e commitando.

Obs: O commit foi apenas para a branch selecionada e não na MASTER.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git status
On branch teste_criacao
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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git commit -a -m "funcionalidade executada"
[teste_criacao df42f89] funcionalidade executada
1 file changed, 1 insertion(+), 1 deletion(-)
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git status
On branch teste_criacao
nothing to commit, working tree clean
PS D:\TDS\UC10 - versionando codigos\repo\1_branches>
```

UNINDO BRANCHES

O código de dois branches distintos podem ser unidos pelo comando **git merge <nome>**.

É por meio deste comando que recebemos as atualizações de outros desenvolvedores.

index.html X

```
1 _branches > <> index.html > html > body
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
7    <title>Document</title>
8  </head>
9  <body>
10   <p>Trabalhando com branches</p>
11   <p>levando modificações para a branch master</p>
12  </body>
13 </html>
```

Atualizando o arquivo index.html e commitando para a branch teste_criacao.

PROBLEMS OUTPUT **TERMINAL** DEBUG CONSOLE

```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git branch
primeiro_branch
* teste_criacao
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git status
On branch teste_criacao
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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git commit -a -m "Comitando atualizações"
[teste_criacao 4b391aa] Comitando atualizações
1 file changed, 1 insertion(+)
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git status
On branch teste_criacao
nothing to commit, working tree clean
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git merge teste_criacao
Updating 4ffb25a..4b391aa
Fast-forward
 index.html | 3 ++-
1 file changed, 2 insertions(+), 1 deletion(-)
```

Mudando para a branch Master e levando as atualizações feitas na branch teste_criacao.

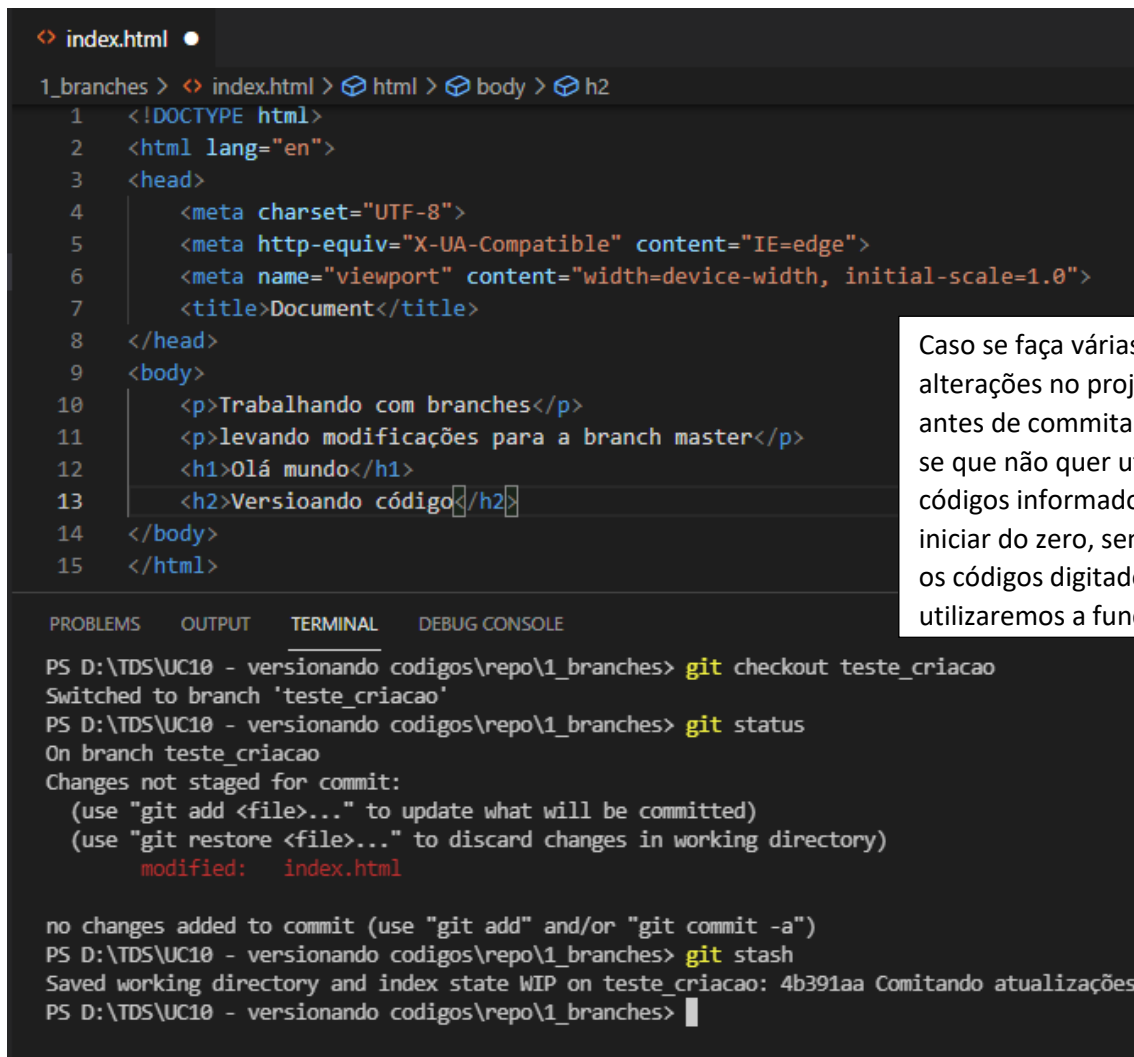
Obs: Também posso fazer merge entre outras branches, sem ser apenas com a Master.

STASH

Podemos salvar as modificações atuais para prosseguir com uma outra abordagem de solução e não perder o código.

O comando para esta ação é o **git stash**.

Após o comando o branch será setado para a sua versão de acordo com o repo.



```
<> index.html ●
1_branches > <> index.html > html > body > h2
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
10     <p>Trabalhando com branches</p>
11     <p>levando modificações para a branch master</p>
12     <h1>Olá mundo</h1>
13     <h2>Versioando código</h2>
14 </body>
15 </html>
```

Caso se faça várias alterações no projeto e antes de commitar, verifica-se que não quer utilizar os códigos informados e sim iniciar do zero, sem perder os códigos digitados, utilizaremos a função stash.

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git checkout teste_criacao
Switched to branch 'teste_criacao'
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git status
On branch teste_criacao
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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git stash
Saved working directory and index state WIP on teste_criacao: 4b391aa Comitando atualizações
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> 
```

RECUPERANDO STASH

Podemos verificar as stashes criadas pelo comando **git stash list** e também podemos recuperar a stash com o comando **git stash <nome>**.

Desta maneira podemos continuar de onde paramos com os arquivos adicionados à stash.

Verificando a lista de stashes

```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git stash list
stash@{0}: WIP on teste_criacao: 4b391aa Comitando atualizações
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git stash apply 0
On branch teste_criacao
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:   index.html
```

Buscando as atualizações da stash 0.

```
no changes added to commit (use "git add" and/or "git commit -a")
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git stash list
stash@{0}: WIP on teste_criacao: 4b391aa Comitando atualizações
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git stash show -p 0
diff --git a/index.html b/index.html
index 4d43b67..3e478e1 100644
--- a/index.html
+++ b/index.html
@@ -9,5 +9,7 @@
<body>
  <p>Trabalhando com branches</p>
  <p>levando modificações para a branch master</p>
+  <h1>Olá mundo</h1>
+  <h2>Versioando código</h2>
</body>
</html>
\ No newline at end of file
```

Verificando quais são as atualizações da stash 0.

REMOVENDO A STASH

Para limpar totalmente as stashes de um branch, utilizamos o comando `git statsh clear`.

Caso seja necessário deletar uma stash específica, utilizamos o comando `git stash drop <nome>`.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git stash
Saved working directory and index state WIP on teste_criacao: 4b391aa Comitando atualizações
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git stash list
stash@{0}: WIP on teste_criacao: 4b391aa Comitando atualizações
stash@{1}: WIP on teste_criacao: 4b391aa Comitando atualizações

PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git stash drop 1
Dropped refs/stash@{1} (6f352a2d3c1ce5f7f60f0f25c1285002db809d48)
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git stash list
stash@{0}: WIP on teste_criacao: 4b391aa Comitando atualizações
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git stash clear
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git stash list
```

UTILIZANDO TAGS

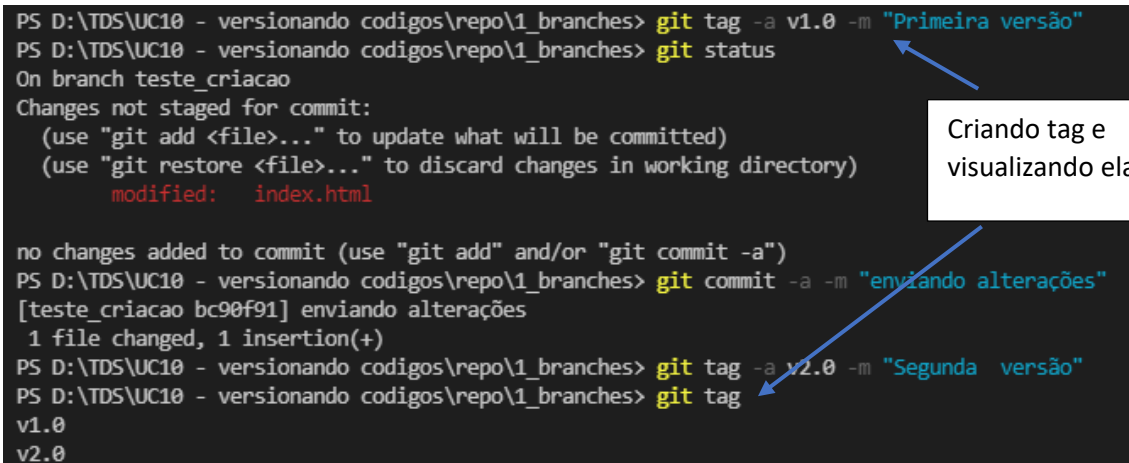
Quando queremos criar marcos no desenvolvimento. Ou seja, serve como um checkpoint de um branch.

É utilizada para demarcar estágios do desenvolvimento de algum recurso.

Podemos criar tags nos branches por meio do comando **git tag -a <nome> -m "<mensagem>"**.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git tag -a v1.0 -m "Primeira versão"
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git status
On branch teste_criacao
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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git commit -a -m "enviando alterações"
[teste_criacao bc90f91] enviando alterações
 1 file changed, 1 insertion(+)
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git tag -a v2.0 -m "Segunda versão"
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git tag
v1.0
v2.0
```



Criando tag e visualizando elas.

VERIFICANDO E ALTERANDO TAGS

Podemos verificar uma tag como comando **git show <nome_da_tag>**.

Podemos trocar de tags com o comando **git checkout <nome_da_tag>**.

Desta forma podemos retroceder ou avançar em checkpoints de um branch.

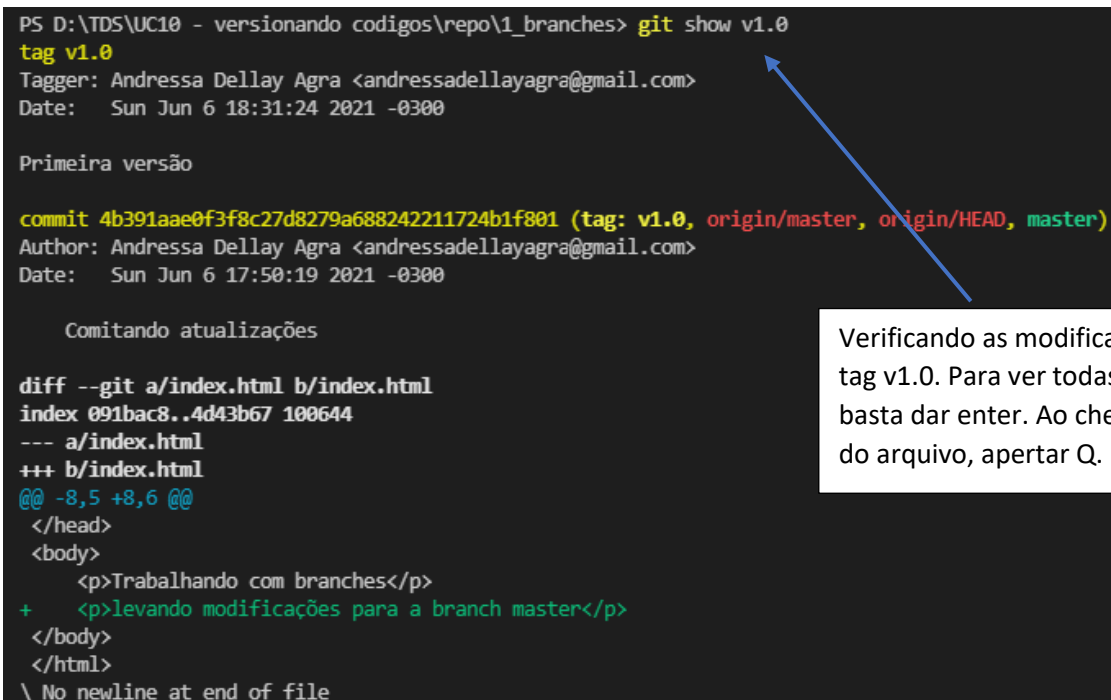
```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git show v1.0
tag v1.0
Tagger: Andressa Dellay Agra <andressadellayagra@gmail.com>
Date:   Sun Jun 6 18:31:24 2021 -0300

Primeira versão

commit 4b391aae0f3f8c27d8279a688242211724b1f801 (tag: v1.0, origin/master, origin/HEAD, master)
Author: Andressa Dellay Agra <andressadellayagra@gmail.com>
Date:   Sun Jun 6 17:50:19 2021 -0300

    Comitando atualizações

diff --git a/index.html b/index.html
index 091bac8..4d43b67 100644
--- a/index.html
+++ b/index.html
@@ -8,5 +8,6 @@
 </head>
 <body>
   <p>Trabalhando com branches</p>
+  <p>levando modificações para a branch master</p>
 </body>
 </html>
\ No newline at end of file
```



Verificando as modificações feitas na tag v1.0. Para ver todas as alterações basta dar enter. Ao chegar no final do arquivo, apertar Q.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git checkout v1.0
Previous HEAD position was e9bca40 Alterando h3
HEAD is now at 4b391aa Comitando atualizações
```

Retrocedendo para a versão de tag v1.0.

ENVIANDO E COMPARTILHANDO TAGS

As tags podem ser enviadas para o repositório de código, sendo compartilhada entre os desenvolvedores.

O comando é **git push origin <nome>**

Se quiser enviar mais tags o comando é **git push origin --tags**.

Enviando para o repositório uma nova tag e não um branch através do push.

```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git tag
v1.0
v2.0
v3.0
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git push origin v2.0
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 495 bytes | 495.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/andressadellay/versionamentodesoftware.git
 * [new tag]          v2.0 -> v2.0
PS D:\TDS\UC10 - versionando codigos\repo\1_branches>
```

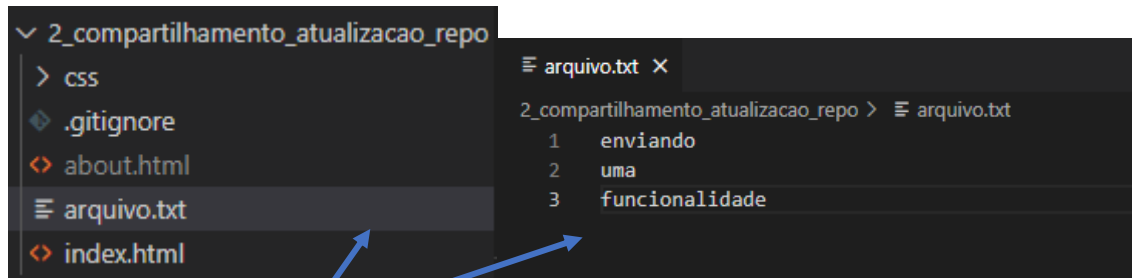
```
PS D:\TDS\UC10 - versionando codigos\repo\1_branches> git push origin --tags
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 617 bytes | 617.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/andressadellay/versionamentodesoftware.git
 * [new tag]          v1.0 -> v1.0
 * [new tag]          v3.0 -> v3.0
PS D:\TDS\UC10 - versionando codigos\repo\1_branches>
```

Enviando para o repositório todas as tags e não um branch através do push.

ENCONTRANDO BRANCHES

Branches são criadas o tempo inteiro pelo time de desenvolvimento, por isso nem sempre nosso git os mapea.

Para fazer o reconhecimento de todas as branches e tags que foram criadas por outros parceiros de equipe, utilizamos o comando **git fetch**.



Para testar o comando `git fetch -a`, criar um arquivo novo em um dos projetos e commitar

Cria uma branch denominada `funcionalidade_a` e muda para ela, para adicionar novo arquivo e commitar.

```
PS D:\TDS\UC10 - versionando codigos\repo\2_compartilhamento_atualizacao_repo> git checkout -b "funcionalidade_a"
Switched to a new branch 'funcionalidade_a'

PS D:\TDS\UC10 - versionando codigos\repo\2_compartilhamento_atualizacao_repo> git add arquivo.txt
PS D:\TDS\UC10 - versionando codigos\repo\2_compartilhamento_atualizacao_repo> git status
On branch funcionalidade_a
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   arquivo.txt

PS D:\TDS\UC10 - versionando codigos\repo\2_compartilhamento_atualizacao_repo> git commit -a -m "enviando arquivo.txt"
[funcionalidade_a f016d46] enviando arquivo.txt
1 file changed, 3 insertions(+)
 create mode 100644 arquivo.txt
PS D:\TDS\UC10 - versionando codigos\repo\2_compartilhamento_atualizacao_repo> git status
On branch funcionalidade_a
nothing to commit, working tree clean
```

```
PS D:\TDS\UC10 - versionando codigos\repo\2_compartilhamento_atualizacao_repo> git push
fatal: The current branch funcionalidade_a has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin funcionalidade_a
```

Diz que temos que enviar a nova branch para dentro do repositório. Basta copiar o comando (selecionada a linha inteira e clica com o botão direito do mouse)

Ao executar o comando, a branch é enviada para dentro do repositório.

```
PS D:\TDS\UC10 - versionando codigos\repo\2_compartilhamento_atualizacao_repo> git push --set-upstream origin funcionalidade_a
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 322 bytes | 322.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'funcionalidade_a' on GitHub by visiting:
remote:   https://github.com/andressadellay/versionamentodesoftware/pull/new/funcionalidade_a
remote:
To https://github.com/andressadellay/versionamentodesoftware.git
 * [new branch]   funcionalidade_a -> funcionalidade_a
Branch 'funcionalidade_a' set up to track remote branch 'funcionalidade_a' from 'origin'.
```

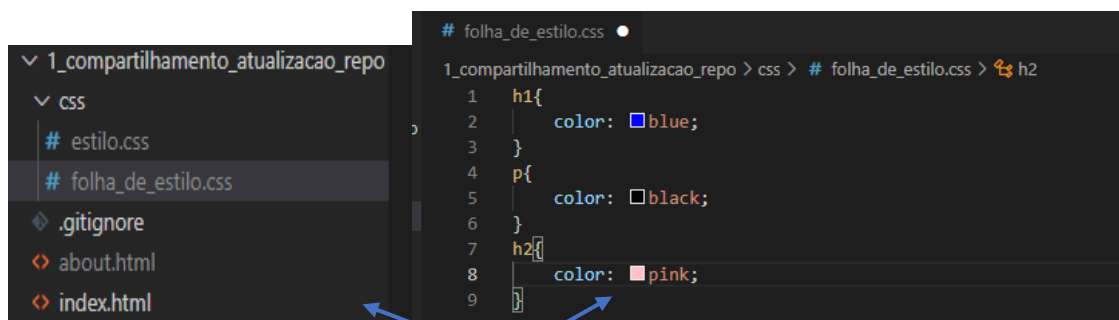
```
PS D:\TDS\UC10 - versionando codigos\repo\1_compartilhamento_atualizacao_repo> git checkout funcionalidade_a
error: pathspec 'funcionalidade_a' did not match any file(s) known to git
PS D:\TDS\UC10 - versionando codigos\repo\1_compartilhamento_atualizacao_repo> git fetch -a
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), 302 bytes | 1024 bytes/s, done.
From https://github.com/andressadellay/versionamentodesoftware
 * [new branch]   funcionalidade_a -> origin/funcionalidade_a
PS D:\TDS\UC10 - versionando codigos\repo\1_compartilhamento_atualizacao_repo> git checkout funcionalidade_a
Switched to a new branch 'funcionalidade_a'
Branch 'funcionalidade_a' set up to track remote branch 'funcionalidade_a' from 'origin'.
```

Após o reconhecimento de todas as branches, é possível fazer um git checkout para a branch desejada, neste caso, funcionalidade_a, onde teremos, também o arquivo.txt.

Se dentro da outra pasta (1_compartilhamento_atualizacao_repo), tentar mudar de branch, o mesmo não o vai reconhecer. Para isso é necessário utilizar o comando git fetch -a.

RECEBENDO ALTERAÇÕES

O comando **git pull** serve para recebermos atualizações do repositório remoto.



No projeto 1_compartilhamento_atualizacao_repo, faço modificações dentro do arquivo folha_de_estilo.css.

```

PS D:\TDS\UC10 - versionando codigos\repo\1_compartilhamento_atualizacao_repo> git status
On branch funcionalida_cc
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:   css/folha_de_estilo.css

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\TDS\UC10 - versionando codigos\repo\1_compartilhamento_atualizacao_repo> git branch
* funcionalida_cc
  funcionalidade_a
  master
PS D:\TDS\UC10 - versionando codigos\repo\1_compartilhamento_atualizacao_repo> git commit -a -m "enviando atualização css"
[funcionalida_cc 5fcaaab] enviando atualização css
1 file changed, 9 insertions(+)
PS D:\TDS\UC10 - versionando codigos\repo\1_compartilhamento_atualizacao_repo> git status
On branch funcionalida_cc
nothing to commit, working tree clean

```

Após modificação do estilo.css (em 1_compartilhamento_atualizacao_repo) e após ter criado uma nova branch, é verificado o status e feito commit para a branch "funcionalidade_cc"

```

PS D:\TDS\UC10 - versionando codigos\repo\1_compartilhamento_atualizacao_repo> git push
fatal: The current branch funcionalida_cc has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin funcionalida_cc

PS D:\TDS\UC10 - versionando codigos\repo\1_compartilhamento_atualizacao_repo> git push --set-upstream origin funcionalida_cc
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 414 bytes | 207.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'funcionalida_cc' on GitHub by visiting:
remote:   https://github.com/andressadellay/versionamentodesoftware/pull/new/funcionalida_cc
remote:
To https://github.com/andressadellay/versionamentodesoftware.git
 * [new branch]      funcionalida_cc -> funcionalida_cc
Branch 'funcionalida_cc' set up to track remote branch 'funcionalida_cc' from 'origin'.

```

Envio o novo branch para o repositório.

Será necessário trocar para a master

```

PS D:\TDS\UC10 - versionando codigos\repo\2_compartilhamento_atualizacao_repo> git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

```

```

PS D:\TDS\UC10 - versionando codigos\repo\2_compartilhamento_atualizacao_repo> git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 1), reused 4 (delta 1), pack-reused 0
Unpacking objects: 100% (4/4), 374 bytes | 0 bytes/s, done.
From https://github.com/andressadellay/versionamentodesoftware

```

Em 2_compartilhamento_atualizacao_repo, dar um git pull na master, para poder receber todas as atualizações enviadas pelos outros membros da equipe.

```

PS D:\TDS\UC10 - versionando codigos\repo\2_compartilhamento_atualizacao_repo> git merge origin/funcionalida_cc

```

Fazendo com que a master receba as alterações realizadas na branch funcionalida_cc

ENVIANDO ALTERAÇÕES

Comando: **git push** – envia as atualizações para o repositório e envia as atualizações específicas de um branch para outro desenvolvedor.

Quando terminamos uma tarefa, precisamos utilizar o git push para enviar ela ao repositório, por isso é comum dar o commit e após o push.

UNTRACKED: arquivos que precisam ser adicionados com o comando git add .Obs: Após o add, dar commit e push

UNSTAGE: arquivos que precisam ser commitados (apenas) com o comando git commit. Obs: Após dar o comando git push para atualizar o repositório.

UTILIZANDO O REMOTE