



**CURSOS FIC EaD
PROGRAMA NOVOS CAMINHOS**

**Apostila da Disciplina de Infraestrutura de
Desenvolvimento**

Disciplina: Infraestrutura de Desenvolvimento

Junho, 2020.

Sumário

1	Considerações Iniciais	3
1.1	Editores de texto:	3
1.2	IDE	3
1.2.1	As características e ferramentas mais comuns encontradas nos IDEs são:	3
1.2.2	Alguns exemplos de IDEs.....	4
1.3	Framework:	5
1.4	Controle de versão	5
1.5	Páginas Web.....	6
2	Instalação	7
2.1.1	Como baixar e instalar o Sublime Text 3	7
2.1.2	Instalando o plugin LocalizedMenu.....	9
2.1.3	Instalação do Git.....	11
3	Estrutura básica da HTML	12
3.1	Criando um projeto no Sublime	12
3.1.1	Estrutura da página	15
3.1.2	Código Usado no Projeto.....	15
3.1.3	Códigos utilizados na aula Git parte 1	16
4	Uma breve História do Git.....	18
4.1.1	O que é Git?.....	18
4.1.2	Estados do Git.....	19
4.1.3	Vamos praticar!	20
4.1.4	Alterando o arquivo	20
4.1.5	Rastreando alterações no repositório.....	20
4.1.6	Gravando alterações no repositório	21
4.1.7	Verificando histórico de alterações.....	21
4.2	Incluindo novos arquivos ao projeto.....	21
4.2.1	Rastreando arquivos	22
4.2.2	Ignorando arquivos	24
4.2.3	Gravando arquivos no repositório	26
4.2.4	Verificando histórico do repositório	28
4.2.5	Verificando mudanças.....	29
4.2.6	Desfazendo mudanças	37
5	Referencias.....	40

1 Considerações Iniciais

O Objetivo da disciplina de Infraestrutura de desenvolvimento é promover uma introdução ao ambiente de desenvolvimento WEB, bem como trazer alguns conceitos comuns a esse meio.

Quando começamos a estudar desenvolvimento/programação surgem questionamentos como: “por onde devo começar?”, “qual linguagem utilizar?”, “qual ferramenta irei utilizar para escrever meu código?”. À medida que essas questões são respondidas surgem novas, como: “o que são editores de códigos”, “o que são IDEs?”, “como deixar meu código escalável?”. Dentro deste tutorial apresentaremos algumas respostas aos questionamentos iniciais que são comuns a todos que se aventuram por estes caminhos.

Dentro da disciplina de Infraestrutura de Desenvolvimento iniciaremos apresentação de ferramentas que possibilitam a criação de projetos que darão origem as primeiras páginas *web*. Para acompanhar as etapas de desenvolvimento também será apresentada ferramenta de controle de versão que possibilita um melhor gerenciamento de cada etapa do referido projeto.

1.1 Editores de texto:

Como o próprio nome já diz, editores de textos são softwares que auxiliam na escrita dos códigos, os editores mais comuns como Microsoft Word ou notpad, tem por objetivo auxiliar em atividades do dia-a-dia como trabalhos de escola, anotações, lembretes etc. Os editores de texto no conceito de programação têm por objetivo oferecer algumas funcionalidades para o desenvolvimento de software e produtividade do programador. Exemplos: Sublime Text e NotPad++.

1.2 IDE

Ambiente de Desenvolvimento Integrado (*Integrated Development Environment*), é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar o processo. Comumente um IDE reúne ferramentas que possibilitam ao desenvolvedor a implementação de projetos mais complexos com linguagens compiladas ou linguagens em que o código escrito precisa ser transformado em uma outra linguagem “intermediária”.

1.2.1 As características e ferramentas mais comuns encontradas nos IDEs são:

- **Editor** - edita o código-fonte do programa escrito na(s) linguagem(ns) suportada(s) pela IDE;
- **Compilador** (*compiler*) - compila o código-fonte do programa, editado em uma linguagem específica e a transforma em linguagem de máquina;
- **Linker** - liga (*linka*) os vários "pedaços" de código-fonte, compilados em linguagem de máquina, em um programa executável que pode ser executado em um computador ou outro dispositivo computacional;
- **Depurador** (*debugger*) - auxilia no processo de encontrar e corrigir defeitos no código-fonte do programa, na tentativa de aprimorar a qualidade de software;
- **Modelagem de dados** (*modeling*) - criação do modelo de classes, objetos, interfaces, associações e interações dos artefatos envolvidos no software com o objetivo de solucionar as necessidades-alvo do software final;
- **Geração de código** - característica mais explorada em Ferramentas CASE, a geração de código também é encontrada em IDEs, contudo com um escopo mais direcionado a *templates* de código comumente utilizados para solucionar problemas rotineiros. Todavia, em conjunto com ferramentas de modelagem, a geração pode gerar praticamente todo o código-fonte do programa com base no modelo proposto, tornando muito mais rápido o processo de desenvolvimento e distribuição do software;
- **Distribuição** (*deploy*) - auxilia no processo de criação do instalador do software, ou outra forma de distribuição, seja discos ou via internet;
- **Testes Automatizados** (automated tests) - realiza testes no software de forma automatizada, com base em scripts ou programas de testes previamente especificados, gerando um relatório, assim auxiliando na análise do impacto das alterações no código-fonte. Ferramentas deste tipo mais comuns no mercado são chamadas robôs de testes;
- **Refatoração** (*refactoring*) - consiste na melhoria constante do código-fonte do software, seja na construção de código mais otimizado, mais limpo e/ou com melhor entendimento pelos envolvidos no desenvolvimento do software. A refatoração, em conjunto com os testes automatizados, é uma poderosa ferramenta no processo de erradicação de "bugs", tendo em vista que os testes "garantem" o mesmo comportamento externo do software ou da característica sendo reconstruída.

1.2.2 Alguns exemplos de IDEs

IntelliJ IDEA- IDE da JetBrains para desenvolvimento em diversas linguagens, principalmente JAVA.

Android Studio - IDE oficial da Google para desenvolvimento na plataforma Android;

Arduino - IDE para microcontroladores linguagem wiring com bibliotecas em C.

Eclipse - Gera código Java (através de plugins, o Eclipse suporta muitas outras linguagens como Python e C / C++);

Netbeans - Gera código Java (e suporta muitas outras linguagens como PHP, Python e C / C++);

Visual Studio - Gera código para Framework.NET, suportando linguagens como Visual Basic.NET, C#, C++, J# e outras compatíveis com.NET;

1.3 Framework:

Usar frameworks é algo que já faz parte do dia-a-dia da maioria dos desenvolvedores, especialmente de quem trabalha com um grande número de projetos que usam funções similares. Afinal, a possibilidade de reutilizar códigos com poucas alterações ajuda a poupar tempo.

Um framework, de modo geral, é um modelo de códigos já existentes para uma função específica necessária ao desenvolvimento de outros softwares.

Em outras palavras, trata-se de uma ferramenta que une códigos comuns a diversos projetos para que o desenvolvedor não precise programar um código novo para funções que já existem.

Também chamado de arcabouço(estrutura), trata-se de um modelo de códigos, e não de um software a ser executado.

Seu principal objetivo é oferecer determinadas funcionalidades prontas aos programadores, as quais servem de base para o desenvolvimento de novos projetos, gerando, assim, mais produtividades e lucratividade ao economizar tempo e cortar custos.

Basicamente, é um *template* com diversas funções que podem ser usadas pelo desenvolvedor. Com ele, é desnecessário gastar tempo para reproduzir a mesma função em diferentes projetos, auxiliando em um gerenciamento ágil de projetos. Em outras palavras, ele é uma estrutura base, uma plataforma de desenvolvimento, como uma espécie de arcabouço. Ele contém ferramentas, guias, sistemas e componentes que agilizem o processo de desenvolvimento de soluções, auxiliando os especialistas de TI em seus trabalhos.

Exemplos de framework:

- Laravel para PHP
- Django para Python
- Rails para Ruby

1.4 Controle de versão

Sistemas de controle de versão (SCV) permitem controlar as mudanças ocorridas nos arquivos de forma segura, evitando a criação de cópias de forma manual que podem levar a erros no projeto. Durante o desenvolvimento de aplicações naturalmente o código passa por diversas versões, com o uso desses sistemas podemos persistir mudanças no código e voltar atrás quando for necessário.

Na disciplina de Infraestrutura de desenvolvimento será utilizado o Git para versionamento dos arquivos criados durante seu desenvolvimento.

Entre as características do Git, está a criação de históricos, ou seja, o desenvolvedor pode alternar entre as versões criadas tendo a tranquilidade de saber que existe um plano “B” caso as mudanças não deem certo, o histórico possibilita o trabalho em equipe e trabalho em equipe em diferentes ambientes. Os SCV permitem a criação de ramificações onde se iniciam várias versões a partir de um ponto e junção das funcionalidades após finalizadas. Possibilitam ainda a rastreabilidade, essa propriedade tem a função de identificar em que ponto a mudança foi feita e o responsável pela mudança.

Para Chacon e Straub (2014), o controle de versão é um sistema que registra alterações em um arquivo ou conjunto de arquivos ao longo do tempo, para que seja possível a recuperação de versões específicas. O uso de um SCV (Sistema de Controle de Versão) ou VCS (Version Control System), permite reverter os arquivos selecionados de volta para um estado anterior, reverter todo o projeto para um estado anterior, comparar alterações ao longo do tempo, ver quem modificou pela última vez, algo que pode estar causando um problema, quem apresentou um problema, quando causou e mais.

1.5 Páginas Web

Uma página de web ou página web (*web page* ou *webpage*) é uma coleção específica de informações fornecidas por um site e exibidas a um usuário em um navegador web. Um site geralmente consiste em muitas páginas web ligadas de maneira coerente. O nome "página web" é uma metáfora de páginas de papel encadernadas em um livro.

O elemento principal de uma página web é um ou mais arquivos de texto escritos na linguagem HTML (*Hypertext Markup Language*) linguagem de marcação de hipertexto. Muitas páginas web também usam código *JavaScript* para comportamento dinâmico e código CSS (*Cascading Style Sheets*) folhas de estilo em cascata para a aparência e estilização da página. Imagens, vídeos e outros arquivos multimídia também são frequentemente incorporados em páginas web.

2 Instalação

2.1.1 Como baixar e instalar o Sublime Text 3

Baixar e instalar o Sublime Text 3 é um processo bastante simples, independente do sistema operacional.

Para Windows:

Passo 1. [Clique aqui](#) para acessar a página de download do Sublime Text 3 ou acesse:

<https://www.sublimetext.com/3>

Passo 2. Escolha a opção “Windows 64 bits - também disponível como versão portátil”.



Figura 1-sublime

Passo 3: Escolha um diretório para baixar o instalador do Sublime Text 3 e clic em salvar

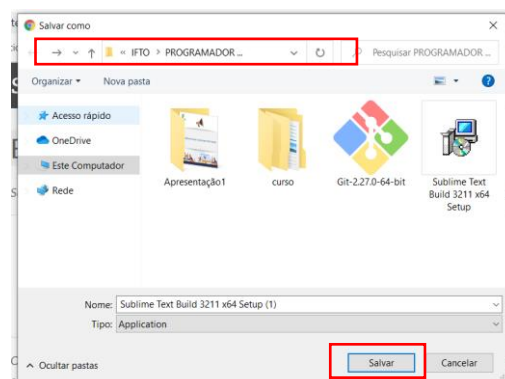


Figura 2-passo da instalação

Passo 4: Execute o instalador com clic duplo

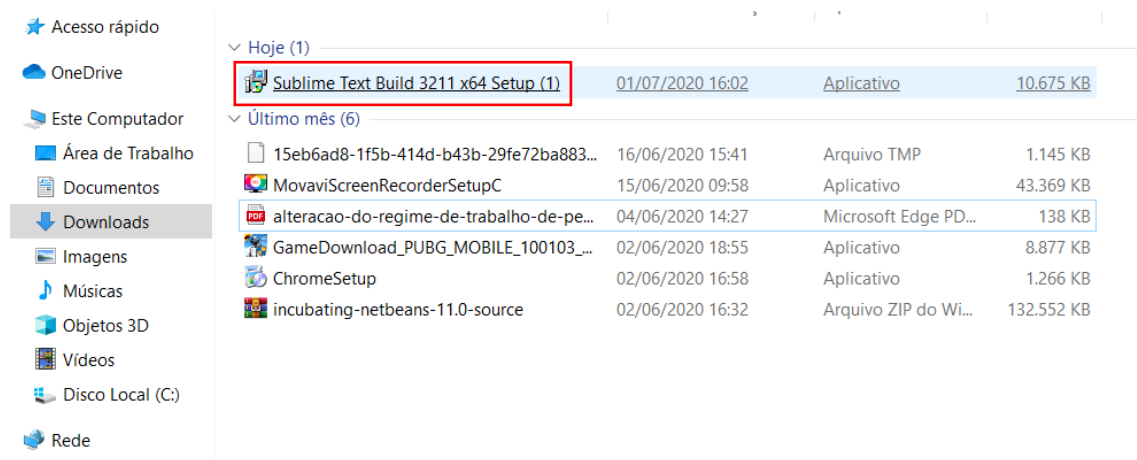
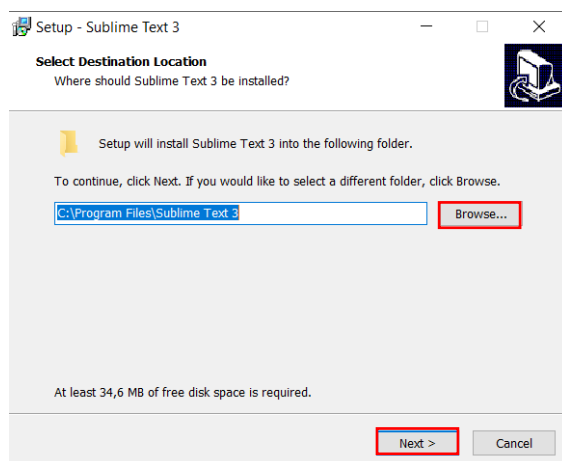


Figura 3- instalador

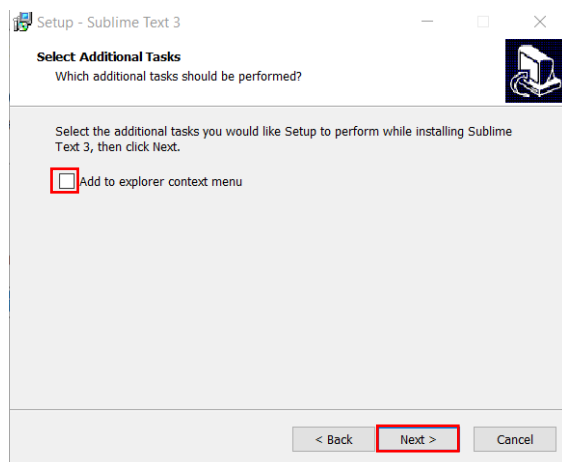
Passo 5: Etapas da Instalação

Tabela 1- Etapas

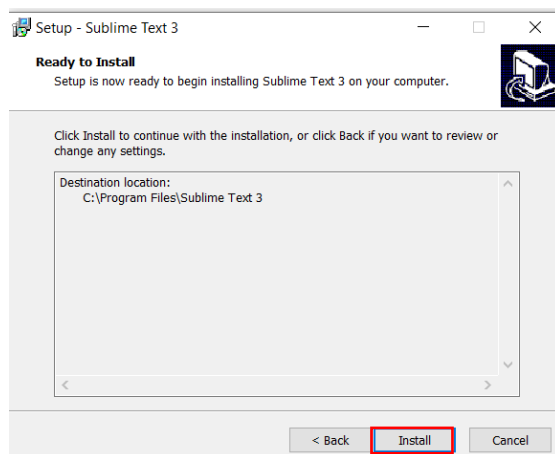


Caso desejar instalar o sublime em outro diretório selecione “Browse..” e depois clic em “Next”.

Caso queira manter no diretório padrão clic “Next”.



Clic na caixa de seleção caso desejar adicionar o sublime ao menu de contexto do *explorer* ou clic em “Next>” para continuar a instalação.



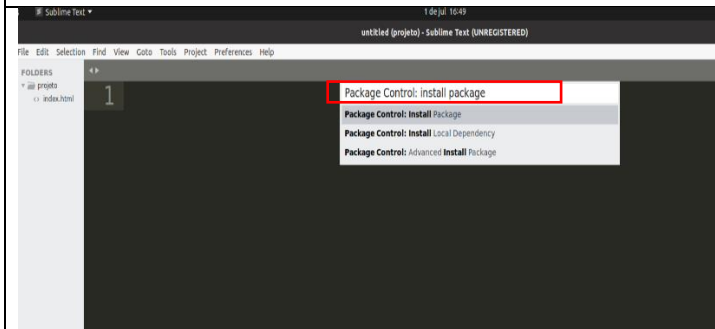
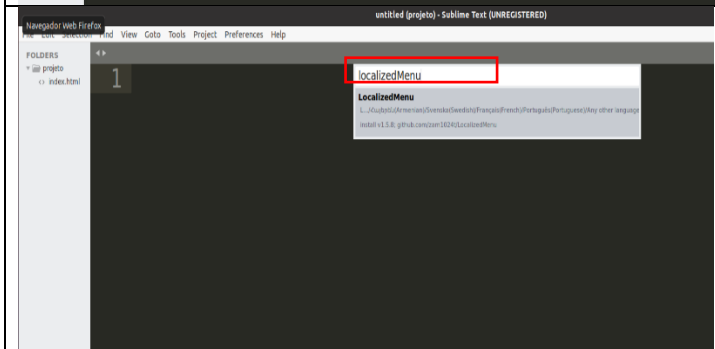
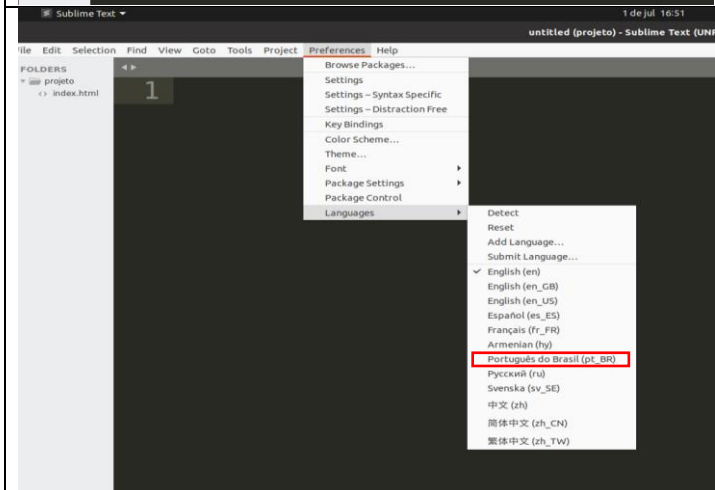
Clic em *install* para continuar a instalação e logo após clic em *finish*.

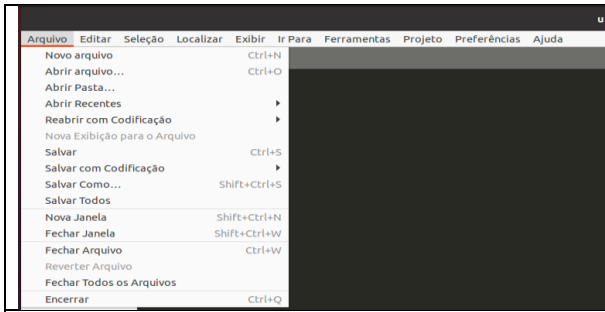
2.1.2 Instalando o plugin LocalizedMenu

Esse plugin permite a tradução dos menus do Sublime.

Tabela 2- Instalação de plugin

	<p>Clic no menu Tools e escolha a opção <i>install Package Control..</i></p>
	<p>No menu Preferences clic na opção <i>Package Control</i></p>

	<p>Em <i>Package Control</i> digite: <i>Install package</i></p>
	<p>Na barra de busca digite nome do pacote que será instalado.</p> <p>e clic na opção de mesmo nome para instalar o plugin.</p>
	<p>Em <i>Preferences</i> aparecerá uma nova opção de menu, selecione o idioma Português do Brasil .</p>

	O resultado.
---	--------------

2.1.3 Instalação do Git

Baixar e instalar o Git é um processo bastante simples, basta baixar o instalador no site <https://git-scm.com/>.

Após baixá-lo execute-o nas etapas que surgirem basta clicar em Next e na penúltima tela do instalador clicar em *Install*.

Após esse processo abra o prompt de comandos do Windows (CMD) e digitar “git version” (sem aspas), conforme imagem abaixo:

Passo 1: Digite cmd na barra de busca e abra o prompt de comandos

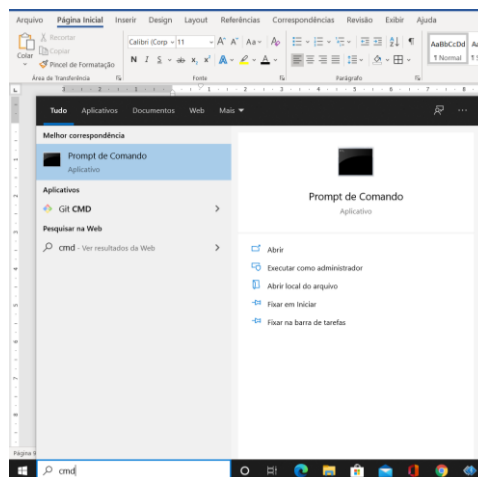


Figura 4- Abrindo CMD

Passo 2: Digite git version no prompt de comandos e caso a instalação tenha sido bem sucedida o prompt exibirá a versão do Git instalado em sua máquina.

```
Prompt de Comando
Microsoft Windows [versão 10.0.19041.329]
(c) 2020 Microsoft Corporation. Todos os direitos reservados.

C:\Users\roge2>git version
git version 2.27.0.windows.1

C:\Users\roge2>
```

Figura 5- Verificando a versão do Git

3 Estrutura básica da HTML

3.1 Criando um projeto no Sublime

Passo 1: Crie um diretório com o nome de sua preferência. (No nosso exemplo usaremos a pasta projeto que está dentro de curso), conforme imagem abaixo:

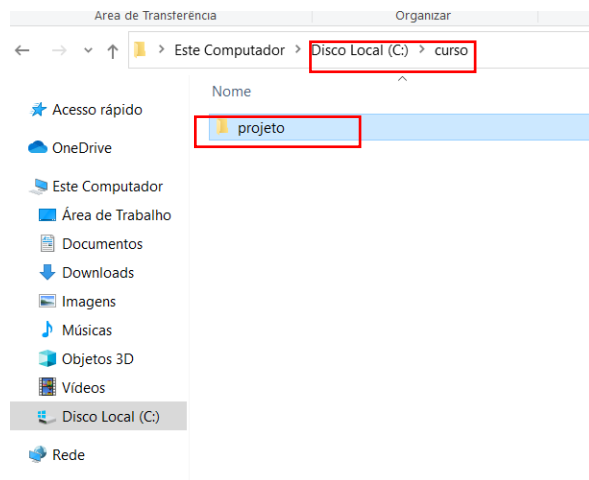


Figura 6- Diretório do projeto

Passo 2: No sublime clic no menu *Project* e em seguida na opção *Add folder to Project*..

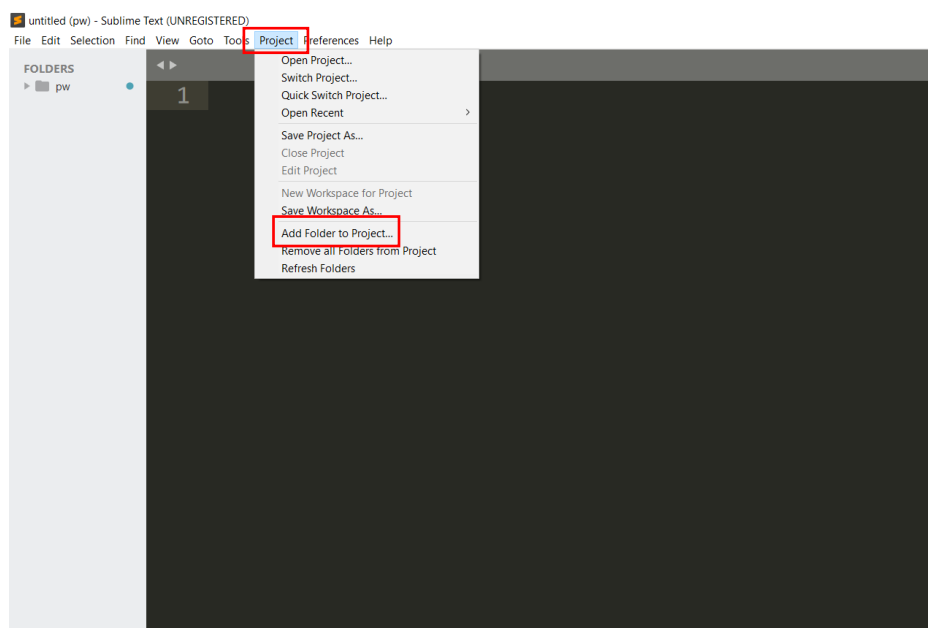


Figura 7- adicionando diretório do projeto ao sublime

Passo 3: Selecione o diretório destinado ao projeto logo após clic em selecionar pasta.

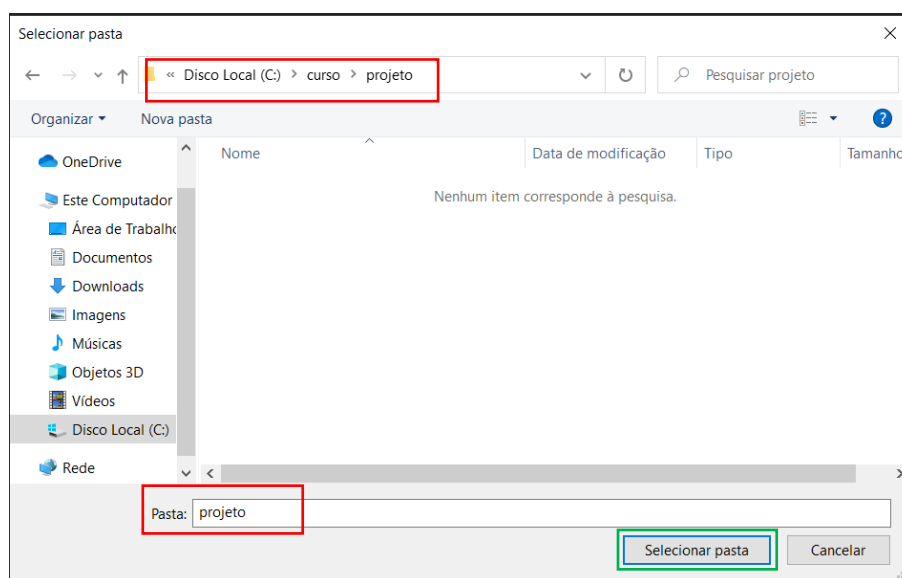


Figura 8- Selecionando diretório

Passo 5: Sublime selecione o diretório do projeto no lado direito da tela em seguida pressione as teclas Ctrl + S na janela que surgir crie um arquivo com o nome de index.html e logo após clic em salvar.

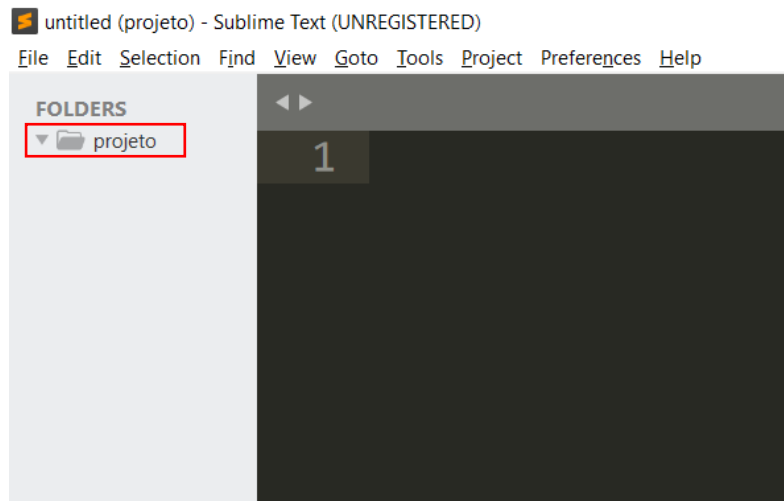


Figura 9- projeto

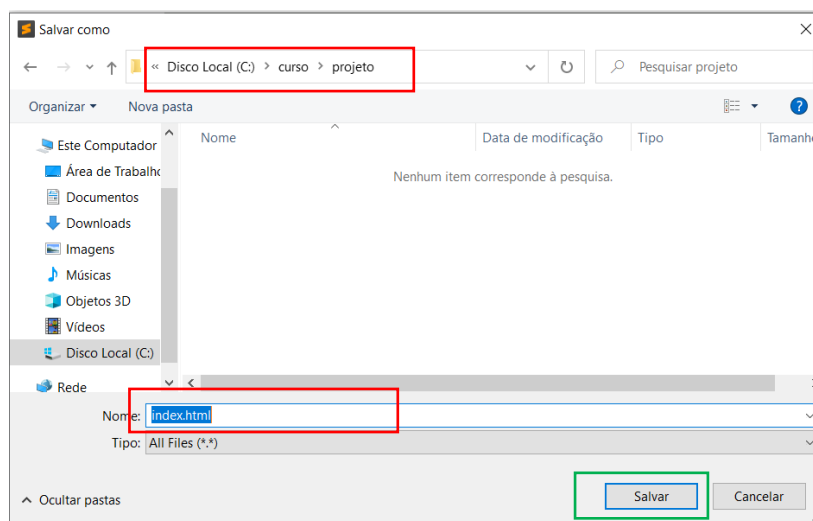


Figura 10- adicionado index.html ao projeto

Passo 6: No editor do sublime digite <htm e pressione as teclas Ctrl + espaço (Auto completar) para mostrar a estrutura do arquivo HTML.

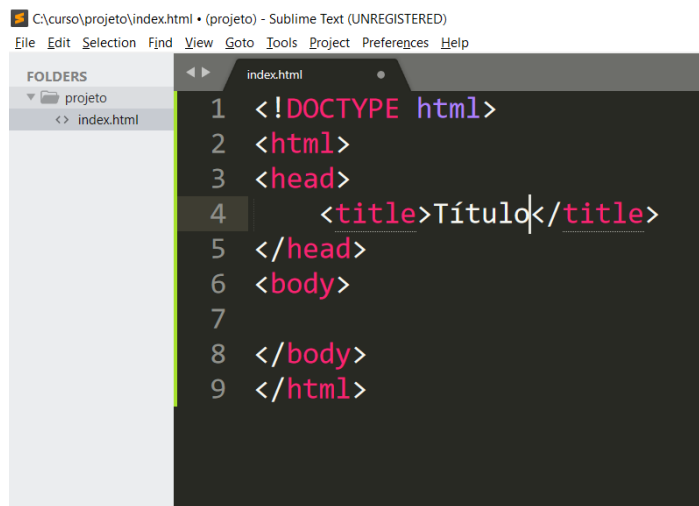


Figura 11- estrutura html

3.1.1 Estrutura da página

A `<!DOCTYPE html>` declaração define que este documento é um documento HTML5

O `<html>` elemento é o elemento raiz de uma página HTML

O `<head>` (cabeça) elemento contém meta informações sobre a página HTML

O `<title>` elemento especifica um título para a página HTML (que é mostrada na barra de título do navegador ou na guia da página)

O `<body>` (corpo) elemento define o corpo do documento e é um contêiner para todo o conteúdo visível, como títulos, parágrafos, imagens, hiperlinks, tabelas, listas, etc.

O `<h1>` elemento define um cabeçalho grande

O `<p>` elemento define um parágrafo

3.1.2 Código Usado no Projeto

```
<!DOCTYPE html>

<html>

<head>

  <title>Minha segunda página</title>

</head>

<body>

<h1>Olá mundo!</h1>

<h1>IFTO produz 20 mil litros de álcool em gel para doação à comunidade tocantinense</h1>

<h5>Benefício direto</h5>

<h3>Coordenadores falam da importância de contribuir no combate à Covid-19</h3>
```

<p>O Instituto Federal do Tocantins (IFTO), entre tantos objetivos, tem o propósito de contribuir com a transformação de realidades, levando benefício direto à população através da oferta de ensino e do desenvolvimento de pesquisas inovadoras.</p>

<p>Diante do cenário difícil enfrentado pela sociedade durante a pandemia da Covid-19, o IFTO cumpre seu papel social de modo a auxiliar no controle da transmissão do novo coronavírus. Nesse sentido, grupos de servidores e estudantes produziram 20 mil litros de gel de álcool etílico glicerinado 76% GL, que serão distribuídos gratuitamente à comunidade tocantinense, de norte a sul do Estado. A produção foi feita utilizando a infraestrutura, insumos e equipamentos do IFTO e teve o apoio da Secretaria de Educação Profissional e Tecnológica (Setec). Participaram do processo de fabricação as unidades de Araguaína, Araguatins, Gurupi, Paraíso do Tocantins e Palmas.</p>

</body>

</html>

3.1.3 Códigos utilizados na aula Git parte 1

`cd c:\curso\projeto` -> Para acessar o diretório do projeto

`dir` -> Mostra o conteúdo do diretório

```
c:\curso\projeto>dir
O volume na unidade C não tem nome.
O Número de Série do Volume é 44F7-7E80

Pasta de c:\curso\projeto

01/07/2020  18:19    <DIR>          .
01/07/2020  18:19    <DIR>          ..
01/07/2020  18:19                0 index.html
                1 arquivo(s)                0 bytes
                2 pasta(s) 42.492.854.272 bytes disponíveis
```

Figura 12- comando dir

`git init` -> inicia um repositório git na pasta raiz do projeto com o nome .git

```
c:\curso\projeto>git init
Initialized empty Git repository in C:/curso/projeto/.git/
```

Figura 13- comando git init

`git status` -> informa sobre o estado da sua *árvore de trabalho*, informa que existem arquivos não rastreados e não comitados. No caso o `index.html`

```
c:\curso\projeto>git status
On branch master

No commits yet

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)
```

Figura 14- comando git status

`git add <nome_do_arquivo>` -> adiciona os arquivos em um espaço intermediário chamado *stage* para que o arquivo possa ser monitorado pelo Git

```
c:\curso\projeto>git add index.html
```

Figura 15- git add

`git status` -> mostra o arquivo adicionado para ser rastreado pelo Git e que existem mudanças a serem comitadas no repositório o nosso `index.html`

```
c:\curso\projeto>git add index.html

c:\curso\projeto>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
```

Figura 16- git status

`git commit -m` “mensagem indicativa que descreve o que foi feito nessa versão” -> Confirma as alterações no documento e cria uma versão grifada em verde, a mensagem grifada em vermelho.

```
c:\curso\projeto>git commit -m "arquivo index.html criado"
[master (root-commit) 891f001] arquivo index.html criado
1 file changed, 15 insertions(+)
create mode 100644 index.html
```

Figura 17-git commit

Obs.:Caso não seja possível comitar, configure usuário e email.

```
git config --global user.name "seu nome"
git config --global user.email "seuemail"
```

4 Uma breve História do Git

Em 2005, o relacionamento entre a comunidade que desenvolveu o kernel do Linux e a empresa comercial que desenvolveu o BitKeeper quebrou e o status gratuito da ferramenta foi revogado. Isso levou a comunidade de desenvolvimento Linux (em particular Linus Torvalds, o criador do Linux) a desenvolver sua própria ferramenta com base em algumas das lições que aprenderam ao usar o BitKeeper. Alguns dos objetivos do novo sistema eram os seguintes:

- Rapidez
- Design simples
- Forte suporte ao desenvolvimento não linear (milhares de ramos paralelos)
- Totalmente distribuído
- Capaz de lidar com grandes projetos como o kernel Linux com eficiência (velocidade e tamanho dos dados)

Desde o seu nascimento em 2005, o Git evoluiu e amadureceu para ser fácil de usar e ainda manter essas qualidades iniciais. É incrivelmente rápido, muito eficiente em grandes projetos e possui um incrível sistema de ramificação para desenvolvimento não linear.

4.1.1 O que é Git?

O Git é um sistema de controle de versões distribuído criado com o intuito de facilitar o gerenciamento de arquivos, criado em 2005, por Linus Torvalds, o mesmo criador do Kernel do Linux.

Para Almeida e Matsui(2017, p.5), “O Git é um sistema de controle de versões que, pela sua estrutura interna, é uma máquina do tempo extremamente rápida um robô de integração bem competente. [...]”.

A maioria das operações no Git precisa apenas de arquivos e recursos locais para operar geralmente nenhuma informação é necessária em outro computador na sua rede. Como você tem todo o histórico do projeto no seu disco local, a maioria das operações parece quase instantânea. (CHACON; STRAUB, 2014).

Para ter acesso ao histórico do projeto e exibi-lo, ele simplesmente lê diretamente do banco de dados local, trazendo o histórico do projeto quase que instantaneamente.

Após inicializar o Git no repositório é impossível alterar o conteúdo de qualquer arquivo sem que o ele saiba, como forma de identificação o Git utiliza hash SHA-1 que é uma cadeia de

40 caracteres composta por caracteres hexadecimais (0–9 e a – f) e calculada com base no conteúdo de uma estrutura de arquivo ou diretório no Git. Um hash SHA-1 é mais ou menos assim: “24b9da6552252987aa493b52f8696cd6d3b00373” .

4.1.2 Estados do Git

O Git possui três estados principais nos quais seus arquivos podem residir, são eles:

- **Modificado (*modified*):** significa que você alterou o arquivo, mas ainda não o confirmou no seu banco de dados.
- **Palco ou índice (*staged*):** significa que você marcou um arquivo modificado em sua versão atual deixando-o preparado para ser confirmado.
- **Confirmado (*committed*):** significa que os dados são armazenados com segurança no seu banco de dados local.

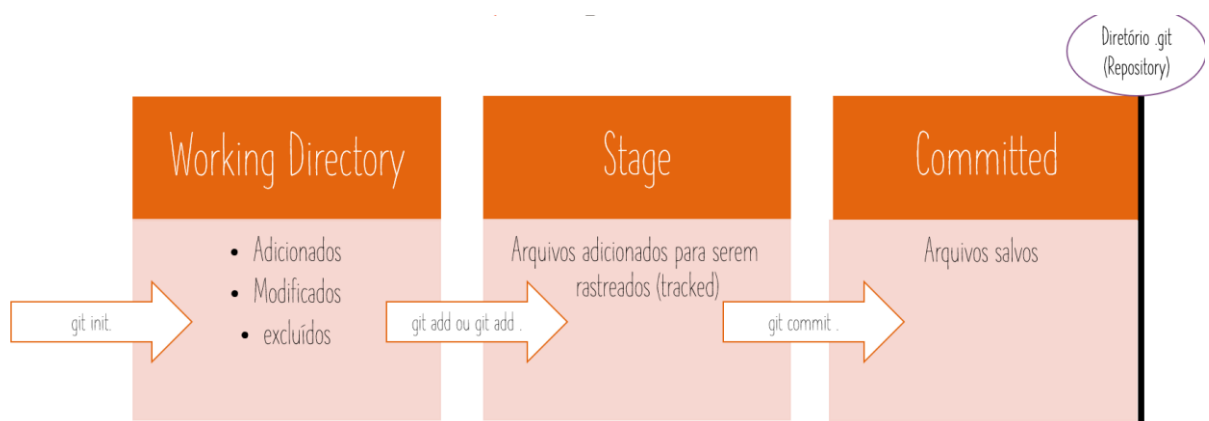


Figura 18- Funcionamento do Git

O Fluxo de trabalho do Git é mais ou menos assim:

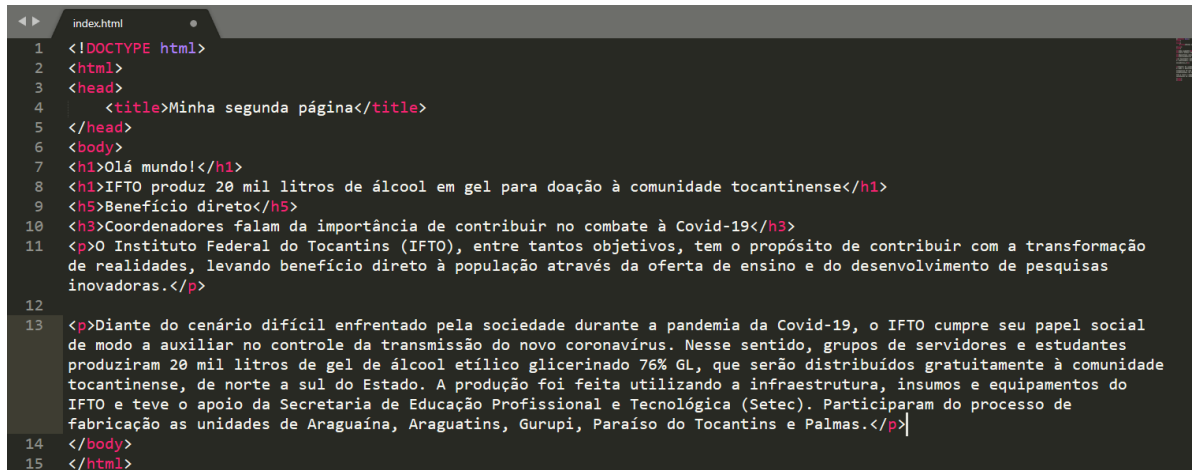
1. Você modifica arquivos na sua árvore de trabalho.
2. Você seleciona seletivamente apenas as alterações que deseja fazer parte do seu próximo ***commit***, o que adiciona **apenas** essas alterações à área de ***stage* ou índice**.
3. Você faz um ***commit***, que pega os arquivos como estão na área de ***stage*** (armazenamento temporário) e armazena os instantâneos permanentemente no diretório Git.

Se uma versão específica de um arquivo estiver no diretório Git, ela será considerada “comitada”. Se foi modificado e foi adicionado ao ***stage***, será considerada preparada. E se foi alterado, mas não foi adicionado ao ***stage***, é considerada modificada.

4.1.3 Vamos praticar!

Agora que verificamos os conceitos iniciais, vamos fazer uma prática de versionamento de arquivo.

Para esta prática daremos continuidade ao projeto criado no item 3.1 desta apostila.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Minha segunda página</title>
5 </head>
6 <body>
7   <h1>Olá mundo!</h1>
8   <h1>IFTO produz 20 mil litros de álcool em gel para doação à comunidade tocaninense</h1>
9   <h5>Benefício direto</h5>
10  <h3>Coordenadores falam da importância de contribuir no combate à Covid-19</h3>
11  <p>O Instituto Federal do Tocantins (IFTO), entre tantos objetivos, tem o propósito de contribuir com a transformação de realidades, levando benefício direto à população através da oferta de ensino e do desenvolvimento de pesquisas inovadoras.</p>
12
13  <p>Diante do cenário difícil enfrentado pela sociedade durante a pandemia da Covid-19, o IFTO cumpre seu papel social de modo a auxiliar no controle da transmissão do novo coronavírus. Nesse sentido, grupos de servidores e estudantes produziram 20 mil litros de gel de álcool etílico glicerinado 76% GL, que serão distribuídos gratuitamente à comunidade tocaninense, de norte a sul do Estado. A produção foi feita utilizando a infraestrutura, insumos e equipamentos do IFTO e teve o apoio da Secretaria de Educação Profissional e Tecnológica (Setec). Participaram do processo de fabricação as unidades de Araguaína, Araguatins, Gurupi, Paraíso do Tocantins e Palmas.</p>
14 </body>
15 </html>
```

Figura 19-Estrutura atual do index.html

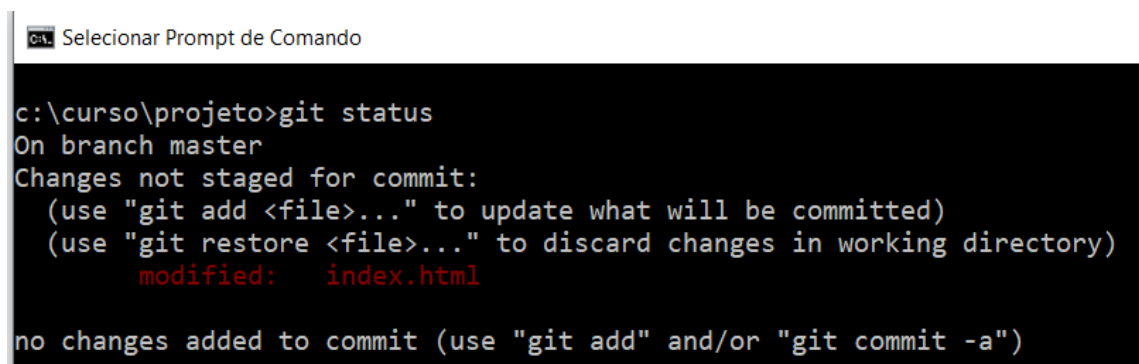
4.1.4 Alterando o arquivo

Insira mais um parágrafo no arquivo index.html, com o conteúdo:

<p>A coordenadora da regional no Tocantins do MIQCB, Emília Alves, parabeniza o IFTO por essa atitude de procurar o movimento para realizar a doação. Além disso, ela agradece a entrega, uma vez que o acesso gratuito ao álcool em gel é um facilitador no combate à disseminação do novo corona vírus. Ainda na oportunidade, a associação foi convidada a apresentar o trabalho das quebradeiras no Instituto, após este contexto de pandemia.</p>

Salve a alteração no seu editor de texto e em seguida, no seu terminal “cmd” execute o comando `git status` .

Podemos observar que há uma nova mudança para ser rastreada:



```
C:\> Selecionar Prompt de Comando

c:\curso\projeto>git status
On branch 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:   index.html

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

Figura 20 - resposta ao comando git status

4.1.5 Rastreado alterações no repositório

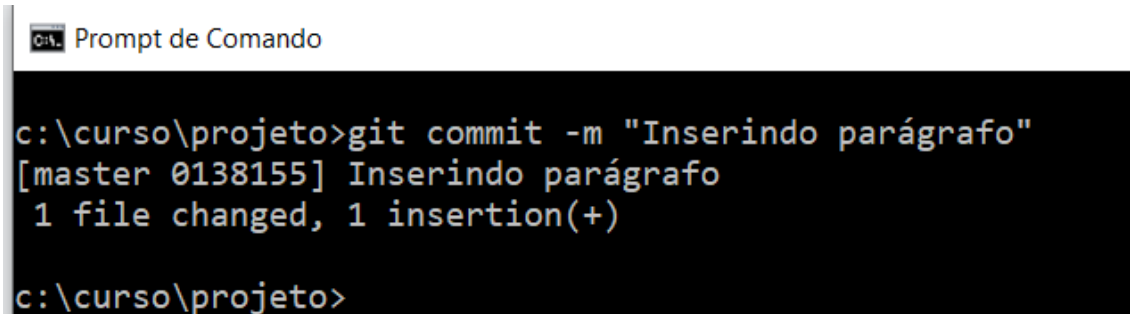
Para rastreamos a modificação, execute o comando `git add index.html`

4.1.6 Gravando alterações no repositório

Com a modificação rastreada, vamos gravá-la no repositório com comando `git add`:

Execute o comando `git commit -m "Inserindo parágrafo"`

Como resposta devemos ter algo parecido com:



```
C:\> Prompt de Comando

c:\curso\projeto>git commit -m "Inserindo parágrafo"
[master 0138155] Inserindo parágrafo
1 file changed, 1 insertion(+)

c:\curso\projeto>
```

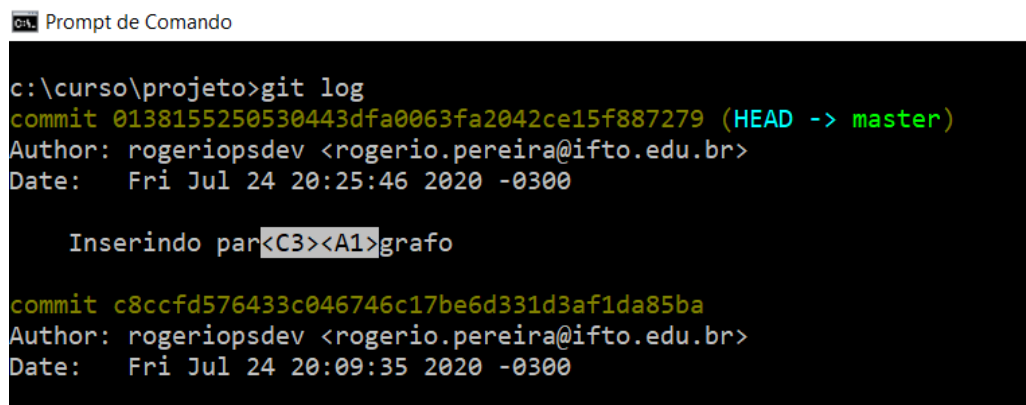
Figura 21- resposta ao comando `git commit`

4.1.7 Verificando histórico de alterações

Para verificarmos o histórico de alterações gravadas, podemos executar o comando `git log`:

No terminal execute: `git log`

Como resposta teremos algo como:



```
C:\> Prompt de Comando

c:\curso\projeto>git log
commit 0138155250530443dfa0063fa2042ce15f887279 (HEAD -> master)
Author: rogeriopsdev <rogerio.pereira@ifto.edu.br>
Date:   Fri Jul 24 20:25:46 2020 -0300

    Inserindo par<C3><A1>grafo

commit c8ccfd576433c046746c17be6d331d3af1da85ba
Author: rogeriopsdev <rogerio.pereira@ifto.edu.br>
Date:   Fri Jul 24 20:09:35 2020 -0300
```

Figura 22 - resposta ao comando `git log`

4.2 Incluindo novos arquivos ao projeto

Vamos inserir um diretório chamado `img` com uma imagem qualquer dentro deste diretório.

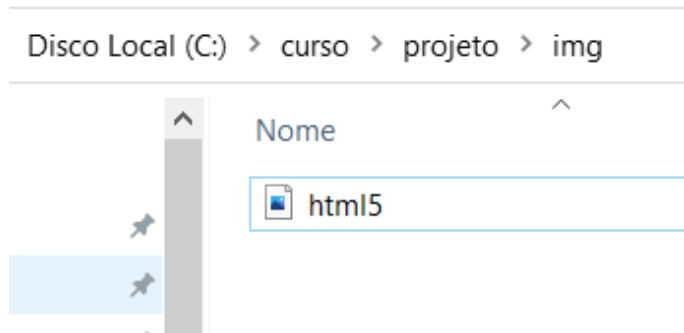


Figura 23 - Adicionando diretório img com imagem ao projeto

4.2.1 Rastreando arquivos

Para verificarmos o estado atual do repositório, executaremos o comando `git status`:

```
git status
```

A resposta deverá ser parecida com essa:

```
C:\> Prompt de Comando

c:\curso\projeto>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    img/

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

Figura 24- Mostrando arquivo não rastreado

Note que o diretório `img/` ainda não está sendo rastreado (*Untracked files*). Para informar que ao Git que o diretório deve ser rastreado executaremos o comando:

```
git add .
```

O Git irá rastrear as mudanças nesse diretório. Para verificar executaremos o comando `git status` novamente.

Teremos a seguinte resposta:

```
C:\> Prompt de Comando

c:\curso\projeto>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   img/html5.jpg
```

Figura 25- Diretório rastreado pelo Git

Observe que o Git agora passou a rastrear o diretório juntamente com a imagem `html5.jpg` e as mudanças estão prontas para serem confirmadas (gravadas) (*Changes to be committed*).

Quando queremos que o Git rastreie um arquivo, executamos o comando `git add`. O Git coloca esse arquivo em uma área especial do repositório, chamada **stage**. Uma vez que o arquivo está nessa área de **stage**, *todas as mudanças nesse arquivo passam a ser examinadas*.

Vamos modificar o arquivo `index.html` acrescentado mais um texto ao corpo da página.

```
<h3>Extensão e Ciência no combate à Covid-19</h3>
```

```
<p>Vale destacar que essa distribuição, iniciada em 16 de junho, está sendo articulada entre as unidades e a Pró-Reitoria de Extensão (Proex), e os produtos serão doados até o término da fabricação.</p>
```

Agora vamos adicionar mais uma imagem ao diretório `img`.



Figura 26- imagem adicionada ao diretório `img`

Para verificar executaremos o comando `git status` novamente.

A saída será:

```
cal. Prompt de Comando

c:\curso\projeto>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   img/html5.jpg

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

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

Figura 27 -Saída do comando `git status`

Note que o index.html continua aparecendo como um novo arquivo rastreado (em verde) em *Changes to be committed:*, indicando que está sendo rastreado e pronto para ser gravado. O index.html aparece mais uma vez em *Changes not staged for commit:*, como modificado (*modified:*).

Isso acontece, uma vez que um arquivo passa a ser rastreado, depois de colocá-lo no *stage* com o comando `git add`, cada mudança é rastreada e deve ser adicionada ao *stage*.

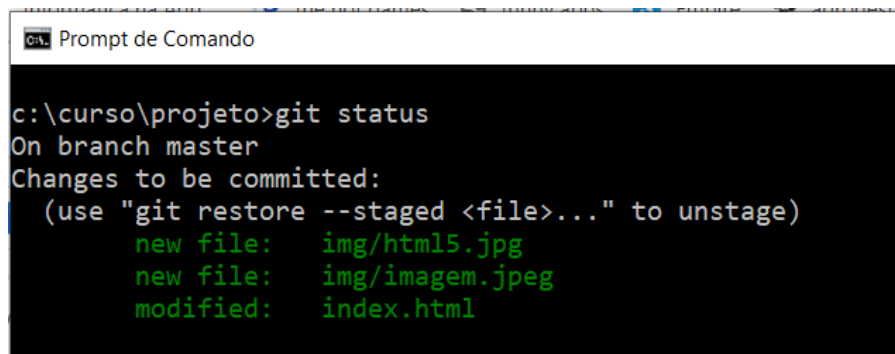
Logo mais abaixo, em *Untracked files:* (arquivos não rastreados) surge a imagem.jpeg que acabamos de adicionar ao diretório img. Nesse caso iremos adicioná-lo ao *stage* para que seja rastreado.

Para que não seja necessário digitar o comando `git add <nomedoarquivo>`, podemos digitar uma das variações desse comando que irá adicionar todas as modificações ao *stage*.

Vamos executar o comando `git add -all`

Execute o comando `git status`

Como saída teremos:



```
Git Prompt de Comando
c:\curso\projeto>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   img/html5.jpg
        new file:   img/imagem.jpeg
        modified:   index.html
```

Figura 28- Saída do comando `git status` após adição de arquivos ao *stage*

4.2.2 Ignorando arquivos

Um arquivo `gitignore` especifica arquivos não rastreados intencionalmente que o Git deve ignorar. Arquivos já rastreados pelo Git não são afetados. Cada linha em um arquivo `gitignore` especifica um padrão.

Suponhamos que temos um arquivo `.txt` que mantemos durante o desenvolvimento. Suponhamos também que temos um documento de planilhas do Excel que utilizo para buscar informações e adicionar ao site. Não faz sentido manter o histórico destes arquivos no Git.

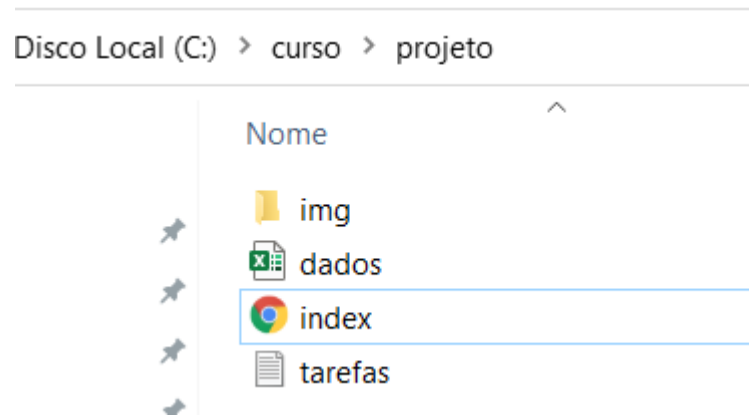


Figura 29- Arquivos que serão ignorados pelo Git

Ao executarmos o comando `git status` os dois arquivos serão mostrados como arquivos não rastreados (em vermelho).

```
cmd Prompt de Comando

c:\curso\projeto>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   img/html5.jpg
    new file:   img/imagem.jpeg
    modified:   index.html

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

Figura 30- Saída do comando `git status`

Para evitar o rastreo desses arquivos o Git tem um mecanismo que permite ignorá-los, basta criarmos um arquivo chamado `.gitignore` no diretório projeto com o seguinte conteúdo:

`dados.xlsx`

`tarefas.txt`

ou

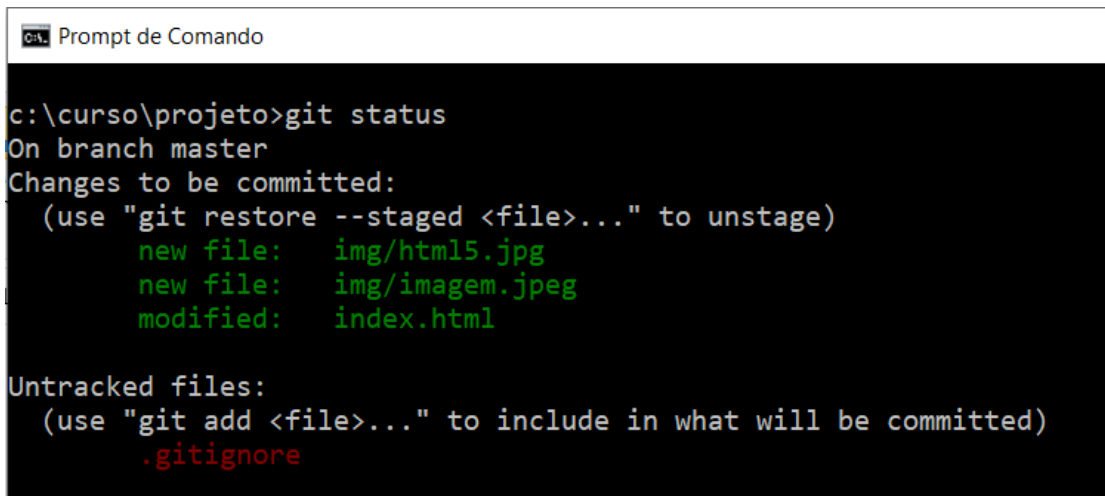
`*.txt`

`*.xlsx`

Com a utilização do asterisco o Git ignorará todos os arquivos que possuem as extensões `.txt` e `.xlsx`.

Execute o comando `git status`:

A saída será algo parecido com:

A terminal window titled 'Prompt de Comando' showing the output of the 'git status' command. The output indicates the current branch is 'master' and lists changes to be committed: 'new file: img/html5.jpg', 'new file: img/imagen.jpeg', and 'modified: index.html'. It also lists untracked files: '.gitignore'.

```
c:\curso\projeto>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   img/html5.jpg
        new file:   img/imagen.jpeg
        modified:   index.html

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

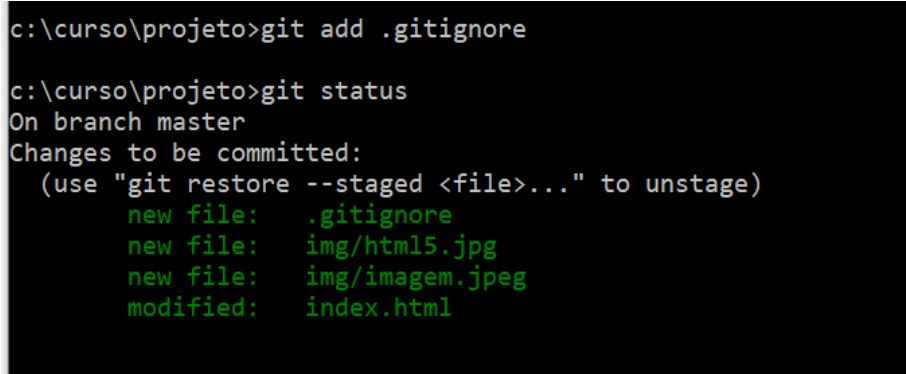
Figura 31- saída do comando `git status` com `.gitignore`

Note que o arquivo `.gitignore` apareceu como não rastreado. É importante que o porquê evoluirá junto com o repositório. Por isso ao criar o `.gitignore`, não esqueça de adicioná-lo ao stage com o comando:

```
git add .gitignore
```

E logo após execute o comando `git status`

A saída do comando será:

A terminal window showing the execution of 'git add .gitignore' followed by 'git status'. The output now includes '.gitignore' as a 'new file' to be committed, along with the previous files.

```
c:\curso\projeto>git add .gitignore

c:\curso\projeto>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore
        new file:   img/html5.jpg
        new file:   img/imagen.jpeg
        modified:   index.html
```

Figura 32 `.gitignore` adicionado ao stage

4.2.3 Gravando arquivos no repositório

Para gravar esses arquivos e alterações definitivamente no repositório devemos utilizar o Git commit:

```
git commit -m "Alterações iniciais"
```

Observe que passamos uma mensagem que descreve as alterações efetuadas via opção `-m`.

Se não passássemos essa opção, seria aberto um editor de texto para informarmos a mensagem.

Depois de executarmos o comando a saída será parecida com:

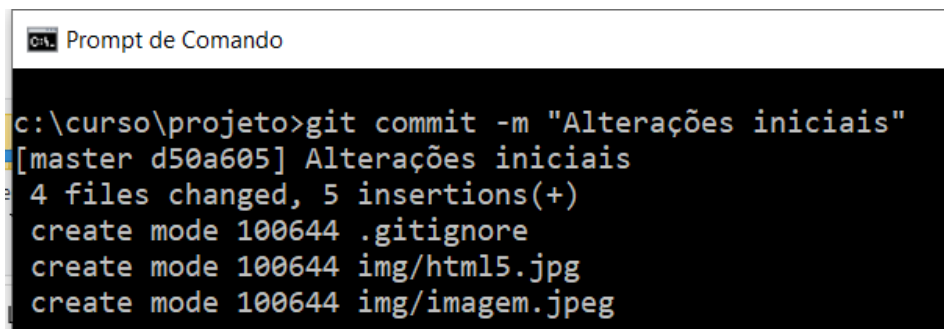
A screenshot of a Windows Command Prompt window titled "Prompt de Comando". The command prompt shows the execution of the command `git commit -m "Alterações iniciais"` in the directory `c:\curso\projeto`. The output is as follows:
`c:\curso\projeto>git commit -m "Alterações iniciais"`
`[master d50a605] Alterações iniciais`
`4 files changed, 5 insertions(+)`
`create mode 100644 .gitignore`
`create mode 100644 img/html5.jpg`
`create mode 100644 img/imagem.jpeg`

Figura 33- Saída do comando git commit

A primeira linha da saída apresenta um código (d50a605 , no caso). Esse código serve de identificador do `commit`, apesar de serem exibidos 7 dígitos o código contém 40 caracteres ao todo.

Execute o comando:

`git status`

A saída será:

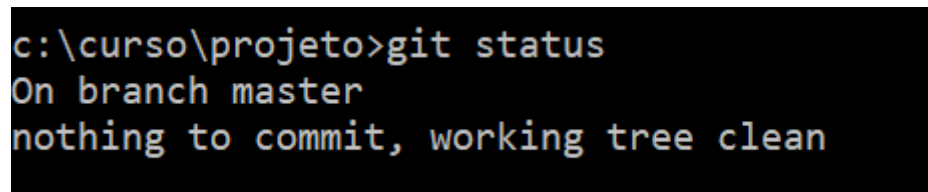
A screenshot of a Windows Command Prompt window showing the execution of the command `git status` in the directory `c:\curso\projeto`. The output is as follows:
`c:\curso\projeto>git status`
`On branch master`
`nothing to commit, working tree clean`

Figura 34-saída do comando git status após gravarmos as alterações

Observe que não há mais nada a ser comitado. O termo *commit* é comum em sistema de controle de versão e significa gravar novos arquivos existentes em um repositório .

4.2.4 Verificando histórico do repositório

Para saber sobre o histórico de mudanças gravadas no repositório, devemos usar o comando:

```
git log
```

A saída será algo assim:

```
      adi<C3><A7><C3><A3>o de par<C3><A1>grafos
commit 891f001d3803c16c87d849a8c916e525e44cd5b5
Author: Rog<C3><A9>rio Rps <roge.23@hotmail.com>
Date:   Wed Jul 1 22:17:46 2020 -0300

    arquivo index.html criado
(END)
```

Figura 35- saída do comando git log

Para sair dos resultados de `git log`, devemos apertar a tecla `q`.

Para mostrar o histórico de `commits` de forma resumida usamos o comando:

```
git log --oneline
```

```
C:\ Prompt de Comando

arquivo index.html criado

c:\curso\projeto>git log --oneline
d50a605 (HEAD -> master) Altera<C3><A7><C3><B5>es iniciais
0138155 Inserindo par<C3><A1>grafo
```

Figura 36- Saída do comando git log --oneline

Para mostrar um número limitado de `commits`, ou seja, digamos que queremos mostrar o resumo das alterações dos últimos 2 `commits`:

```
git log -n 2 --oneline
```

```
c:\curso\projeto>git log -n 2 --oneline
d50a605 (HEAD -> master) Altera<C3><A7><C3><B5>es iniciais
0138155 Inserindo par<C3><A1>grafo

c:\curso\projeto>
```

Figura 37- Saída do comando git log -n2 --oneline

Para mostrarmos o resumo dos arquivos alterados com número de linhas adicionadas e removidas usamos a opção `--stat` para os dois últimos `commits`.

```
git log -n 2 --oneline --stat
```

```
Prompt de Comando

c:\curso\projeto>git log -n 2 --oneline --stat
d50a605 (HEAD -> master) Altera<C3><A7><C3><B5>es iniciais
.gitignore | 2 ++
img/html5.jpg | Bin 0 -> 28706 bytes
img/imagem.jpeg | Bin 0 -> 13479 bytes
index.html | 3 +++
4 files changed, 5 insertions(+)
0138155 Inserindo par<C3><A1>grafo
index.html | 1 +
1 file changed, 1 insertion(+)
```

Figura 38- Saída do comando git log -n 2 --oneline --stat

4.2.5 Verificando mudanças

Em nosso arquivo index.html adicionaremos duas imagens usando as tags:

```
 <!-- na primeira linha do body-->
```

```
 <!-- na última linha do body-->
```

```
6 <body>
7    <!-- na primeira linha do body-->
8   <h1>Olá mundo!</h1>
9   <h1>IFTO produz 20 mil litros de álcool em gel para doação à comunidade tocantinense</h1>
10  <h5>Benefício direto</h5>
11  <h3>Coordenadores falam da importância de contribuir no combate à Covid-19</h3>
12  <p>O Instituto Federal do Tocantins (IFTO), entre tantos objetivos, tem o propósito de contribuir com a transformação
13  de realidades, levando benefício direto à população através da oferta de ensino e do desenvolvimento de pesquisas
14  inovadoras.</p>
15
16  <p>Diante do cenário difícil enfrentado pela sociedade durante a pandemia da Covid-19, o IFTO cumpre seu papel social
17  de modo a auxiliar no controle da transmissão do novo coronavírus. Nesse sentido, grupos de servidores e estudantes
18  produziram 20 mil litros de gel de álcool etílico glicerinado 76% GL, que serão distribuídos gratuitamente à comunidade
19  tocantinense, de norte a sul do Estado. A produção foi feita utilizando a infraestrutura, insumos e equipamentos do
20  IFTO e teve o apoio da Secretaria de Educação Profissional e Tecnológica (Setec). Participaram do processo de
21  fabricação as unidades de Araguaína, Araguaatins, Gurupi, Paraíso do Tocantins e Palmas.</p>
22  <p>A coordenadora da regional no Tocantins do MIQCB, Emilia Alves, parabeniza o IFTO por essa atitude de procurar o
23  movimento para realizar a doação. Além disso, ela agradece a entrega, uma vez que o acesso gratuito ao álcool em gel é
24  um facilitador no combate à disseminação do novo coronavírus. Ainda na oportunidade, a associação foi convidada a
25  apresentar o trabalho das quebradeiras no Instituto, após este contexto de pandemia.</p>
26
27  <h3>Extensão e Ciência no combate à Covid-19</h3>
28  <p>Vale destacar que essa distribuição, iniciada em 16 de junho, está sendo articulada entre as unidades e a
29  Pró-Reitoria de Extensão (Proex), e os produtos serão doados até o término da fabricação.</p>
30   <!-- na última linha do body-->
31 </body>
```

Figura 39- Sugestão de inserção de imagens

No navegador teremos:



Olá mundo!

IFTO produz 20 mil litros de álcool em gel para doação à comunidade tocantinense

Benefício direto

Coordenadores falam da importância de contribuir no combate à Covid-19

O Instituto Federal do Tocantins (IFTO), entre tantos objetivos, tem o propósito de contribuir com a transformação de realidades, levando benefício direto à população através da oferta de ensino e do desenvolvimento de pesquisas inovadoras.

Diante do cenário difícil enfrentado pela sociedade durante a pandemia da Covid-19, o IFTO cumpre seu papel social de modo a auxiliar no controle da transmissão do novo coronavírus. Nesse sentido, grupos de servidores e estudantes produziram 20 mil litros de gel de álcool etílico glicerinado 70% GL, que serão distribuídos gratuitamente à comunidade tocantinense, de norte a sul do Estado. A produção foi feita utilizando a infraestrutura, insumos e equipamentos do IFTO e teve o apoio da Secretaria de Educação Profissional e Tecnológica (Setec). Participaram do processo de fabricação as unidades de Araguaína, Araguatins, Gurupi, Paraíso do Tocantins e Palmas.

A coordenadora da regional no Tocantins do MIQCB, Emília Alves, parabeniza o IFTO por essa atitude de procurar o movimento para realizar a doação. Além disso, ela agradece a entrega, uma vez que o acesso gratuito ao álcool em gel é um facilitador no combate à disseminação do novo coronavírus. Ainda na oportunidade, a associação foi convidada a apresentar o trabalho das queixadeiras no Instituto, após este contexto de pandemia.

Extensão e Ciência no combate à Covid-19

Vale destacar que essa distribuição, iniciada em 16 de junho, está sendo articulada entre as unidades e a Pró-Reitoria de Extensão (Proex), e os produtos serão doados até o término da fabricação.



Figura 40- O que será mostrado no navegador

4.2.5.1 Verificando mudanças ainda não rastreadas

Para revisar mudanças efetuadas, verificando as diferenças entre arquivos comitados e o que foi alterado, usamos o comando:

```
git diff
```

obs.: Aperte a tecla q caso precise sair do resultado do comando anterior.

A saída será semelhante a:

```
</head>
<body>
   <!-- na primeira linha do body-->
  <h1>Olá mundo!</h1>
  <h1>IFTO produz 20 mil litros de álcool em gel para doação à comunidade to
  <h5>Benefício direto</h5>
  <h3>Coordenadores falam da importância de contribuir no combate à Covid-19</h3>
  <p>O Instituto Federal do Tocantins (IFTO), entre tantos objetivos, tem o propósito de contribu
  <h3>Extensão e Ciência no combate à Covid-19</h3>
  <p>Vale destacar que essa distribuição, iniciada em 16 de junho, está sendo artic
   <!-- na última linha do body-->
</body>
```

Figura 41- Saída do comando git diff

Execute o comando `git status`

```
c:\curso\projeto>git status
On branch 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:   img/html5.jpg
        modified:   index.html

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

Figura 42- mudanças não adicionadas ao stage

Execute o comando:

```
git add .
```

Logo após execute o comando:

```
git diff
```

Observe que o comando não mostra nenhuma saída, ou seja, não consegue mais verificar as alterações uma vez que as modificações foram mandadas para o `stage`.

4.2.5.2 Verificando mudanças rastreadas

É possível mostrar as diferenças entre os arquivos na área de `stage` e a última versão que foi comitada, utilizando a opção `--staged`.

Execute o comando :

```
git diff --staged
```

Será exibida a saída uma saída exatamente igual a anterior, sendo que as alterações apenas mudaram para a área de `stage`.

Execute o comando:

```
git commit -m "Adicionando imagens"
```

Como saída do comando teremos algo parecido com:

 Prompt de Comando

```
c:\curso\projeto>git commit -m "adicionando imagens"
[master 370c2d4] adicionando imagens
 2 files changed, 3 insertions(+)
 rewrite img/html5.jpg (92%)
```

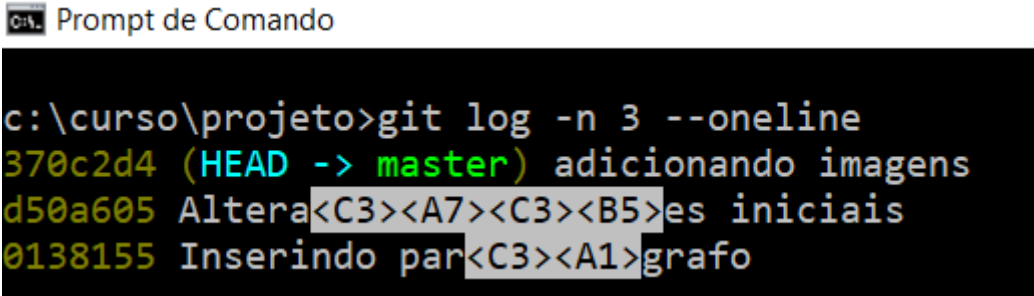
Figura 43- Gravando alterações

4.2.5.3 Verificando mudanças já comitadas

Vamos mostrar os últimos 3 commits de maneira resumida, executando o comando:

```
git log -n 3 --oneline
```

Teremos:



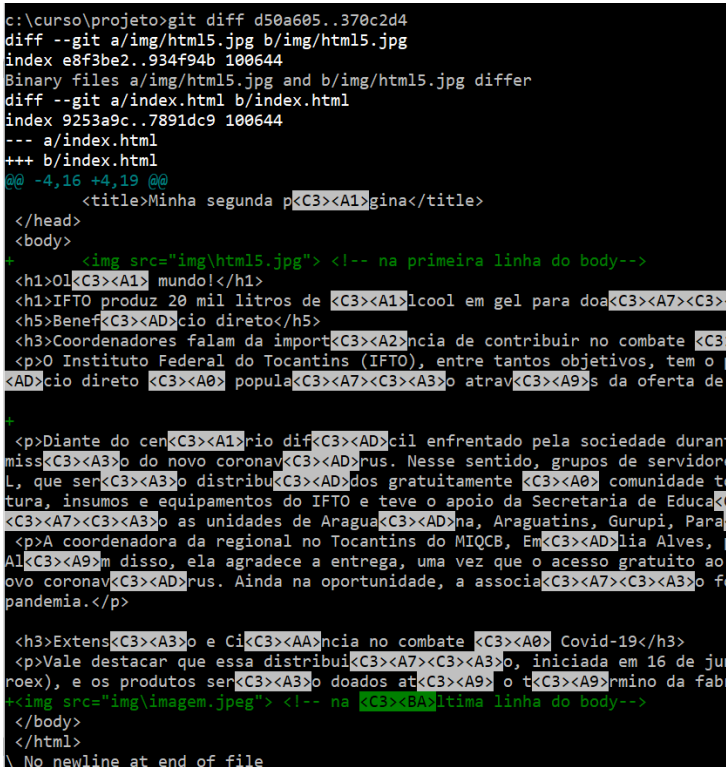
```
Crit. Prompt de Comando
c:\curso\projeto>git log -n 3 --oneline
370c2d4 (HEAD -> master) adicionando imagens
d50a605 Altera<C3><A7><C3><B5>es iniciais
0138155 Inserindo par<C3><A1>grafo
```

Figura 44- Últimos 3 commits

Vamos utilizar o comando `git diff` para verificar as diferenças entre os dois últimos commits, utilizando os códigos desses commits separados por pontos (..):

```
git diff d50a605..370c2d4
```

Teremos algo assim como saída:



```
c:\curso\projeto>git diff d50a605..370c2d4
diff --git a/img/html5.jpg b/img/html5.jpg
index e8f3be2..934f94b 100644
Binary files a/img/html5.jpg and b/img/html5.jpg differ
diff --git a/index.html b/index.html
index 9253a9c..7891dc9 100644
--- a/index.html
+++ b/index.html
@@ -4,16 +4,19 @@
     <title>Minha segunda p<C3><A1>gina</title>
   </head>
   <body>
+    
     <h1>Ol<C3><A1> mundo!</h1>
     <h1>IFT0 produz 20 mil litros de <C3><A1>licool em gel para doa<C3><A7><C3>
     <h5>Benef<C3><AD>cio direto</h5>
     <h3>Coordenadores falam da import<C3><A2>ncia de contribuir no combate <C3>
     <p>O Instituto Federal do Tocantins (IFT0), entre tantos objetivos, tem o p
     <AD>cio direto <C3><A0> popula<C3><A7><C3><A3>o atrav<C3><A9>s da oferta de
+
     <p>Diante do cen<C3><A1>rio dif<C3><AD>cil enfrentado pela sociedade durant
     miss<C3><A3>o do novo coronav<C3><AD>rus. Nesse sentido, grupos de servidores
     L, que ser<C3><A3>o distribu<C3><AD>dos gratuitamente <C3><A0> comunidade t
     tura, insumos e equipamentos do IFT0 e teve o apoio da Secretaria de Educa
     <C3><A7><C3><A3>o as unidades de Aragua<C3><AD>na, Araguatins, Gurupi, Para
     <p>A coordenadora da regional no Tocantins do MIQCB, Em<C3><AD>lia Alves, p
     Al<C3><A9>m disso, ela agradece a entrega, uma vez que o acesso gratuito ao
     ovo coronav<C3><AD>rus. Ainda na oportunidade, a associa<C3><A7><C3><A3>o fo
     pandemia.</p>
+
     <h3>Extens<C3><A3>o e Ci<C3><AA>ncia no combate <C3><A0> Covid-19</h3>
     <p>Vale destacar que essa distribu<C3><A7><C3><A3>o, iniciada em 16 de jun
     roex), e os produtos ser<C3><A3>o doados at<C3><A9> o t<C3><A9>rmino da fabri
+img src="img\imagem.jpeg" <!-- na <C3><BA>ltima linha do body-->
   </body>
 </html>
\ No newline at end of file
```

Figura 45- Verificando mudanças já comitadas

Adicione dois arquivos chamados `principal.css` e `ajustes.css` respectivamente ao diretório projeto. Ficará assim:

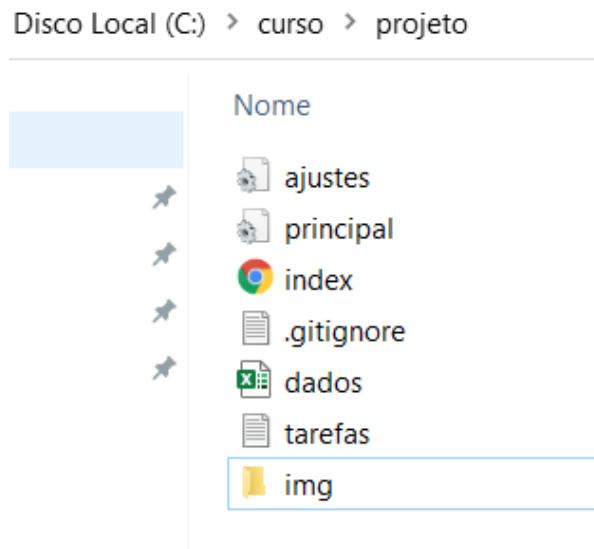


Figura 46- visão geral do diretório

Para adicioná-los ao `stage`, execute o comando:

```
git add .
```

Pronto! Como de costume não será retornado nenhum resultado.

Execute o comando:

```
git status
```

Teremos:

```
c:\curso\projeto>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   ajustes.css
    new file:   principal.css
```

Figura 47- arquivos adicionados ao stage

4.2.5.4 Removendo arquivos

Acabamos percebendo que não precisamos do arquivo `ajustes.css`, porém não basta deletarmos o arquivo `ajustes.css`, precisamos deletá-lo e adicionar a deleção ao `stage`, para só

então efetuar o `commit` no repositório. A remoção e adição podem ser realizadas de uma só vez com o comando:

```
git rm -f ajustes.css
```

Como saída do comando teremos:

```
c:\curso\projeto>git rm -f ajustes.css
rm 'ajustes.css'
```

Figura 48- removendo o arquivo ajustes.css

Adicione um diretório chamado `css` ao diretório `projeto`. Ficará assim:

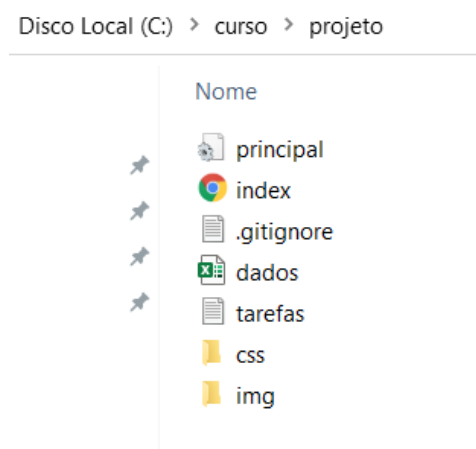


Figura 49-adicionando diretório css

Para adicioná-los ao `stage`, execute o comando:

```
git add .
```

[4.2.5.5 Movendo arquivo](#)

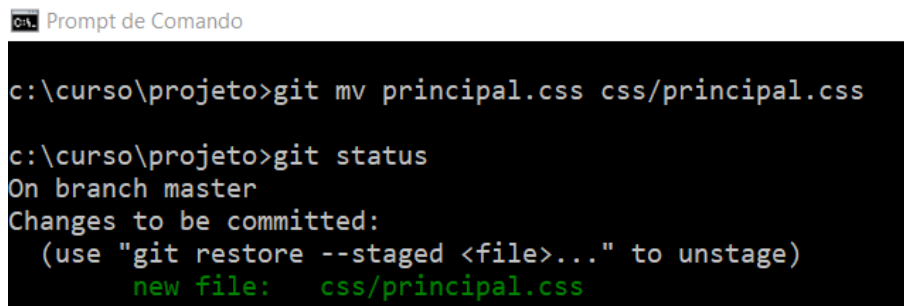
Note que o nosso arquivo `principal.css` está fora do diretório `css`. Para move-lo para dentro do diretório `css` executaremos o comando `git mv`:

```
git mv principal.css css/principal.css
```

Após execute o comando:

```
git status
```

Como saída do comando teremos:



```
Git Prompt de Comando

c:\curso\projeto>git mv principal.css css/principal.css

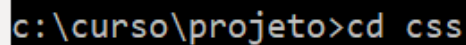
c:\curso\projeto>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   css/principal.css
```

Figura 50- movendo principal.css para o diretório css

4.2.5.6 Renomeando

Para realizar um teste vamos alterar o nome do arquivo principal.css para ajustes.css.

Entre no diretório css, com o comando cd:



```
c:\curso\projeto>cd css
```

Figura 51- mudando de diretório

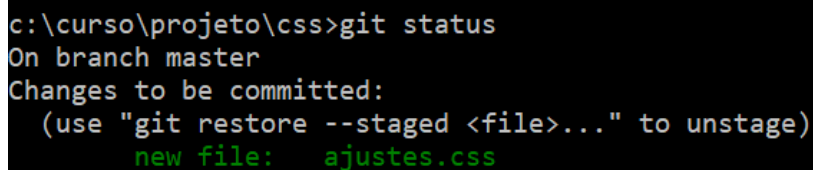
Execute o comando :

```
git mv principal.css ajustes.css
```

Execute o comando:

```
git status
```

Teremos:



```
c:\curso\projeto\css>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   ajustes.css
```

Figura 52-arquivo css renomeado

Para mantermos o padrão de nomenclatura, vamos modificar o nome de ajustes.css para principal.css.

Execute o comando:

```
git mv ajustes.css principal.css
```

Execute o comando:

git status

Teremos:

```
C:\> Prompt de Comando

c:\curso\projeto\css>git mv ajustes.css principal.css

c:\curso\projeto\css>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   principal.css
```

Figura 53- renomeando ajustes.css para principal.css

Volte para o diretório do projeto. Execute o comando:

cd..

Teremos:

```
c:\curso\projeto\css>cd..

c:\curso\projeto>
```

Figura 54- voltando ao diretório projeto

Adicione um título à ao documento indes.html. Texto abaixo:

```
<h2>Título teste</h2><!-- na linha acima da primeira tag img-->
```

Atualize a página no navegador(f5).

Teremos algo parecido com:

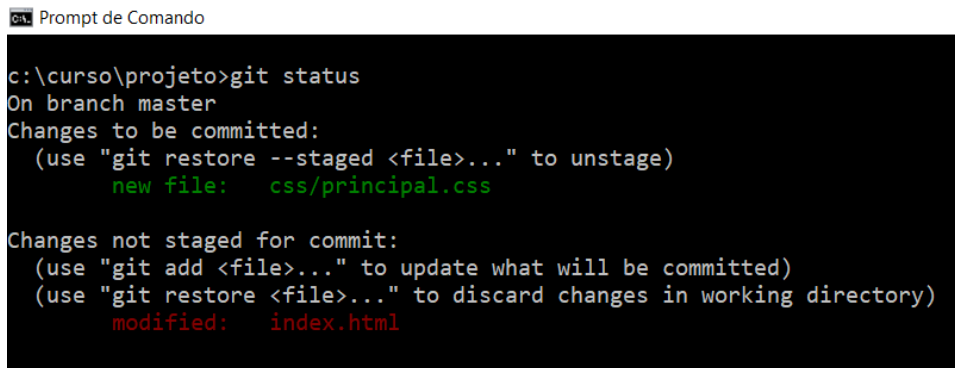


Figura 55- incluindo título teste

No CMD, execute o comando:

```
git status
```

Teremos:



```
C:\> Prompt de Comando

c:\curso\projeto>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   css/principal.css

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
```

Figura 56- modificação não rastreada no index.html

4.2.6 Desfazendo mudanças

4.2.6.1 Desfazendo mudanças não rastreadas

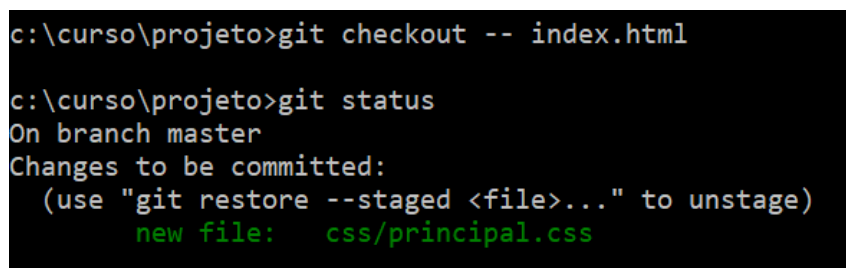
Na **figura 56**, observamos que temos uma modificação no index.html que não foi adicionada ao stage, ou seja, ainda não está rastreada. Para desfazer alterações nesse nível devemos executar o comando:

```
git checkout -- index.html
```

Após, execute o comando:

```
git status
```

Teremos:



```
c:\curso\projeto>git checkout -- index.html

c:\curso\projeto>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   css/principal.css
```

Figura 57- desfazendo alterações não rastreadas

4.2.6.2 Desfazendo mudanças já rastreadas

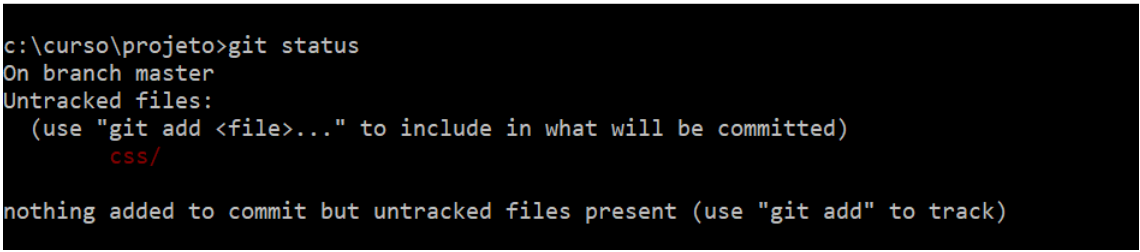
Note que na figura 57, temos um arquivo sendo rastreado (new file: css/principal.css) em verde. Caso já tenha adicionado ao stage uma mudança indesejada podemos voltar atrás com o comando:

```
git reset -- css/principal.css
```

Após, execute o comando:

```
git status
```

Teremos:



```
ca. Prompt de Comando
c:\curso\projeto>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    css/
nothing added to commit but untracked files present (use "git add" to track)
```

Figura 58- O diretório css saiu da área de stage – não rastreado

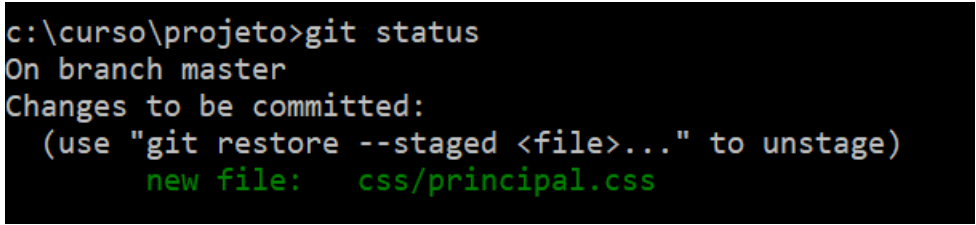
Note que o `css/` saiu da área de `stage` para a área não rastreada. Agora faremos o caminho inverso, execute o comando `git add` par adicionar o diretório `css` ao `stage` novamente:

```
git add .
```

Após, execute o comando:

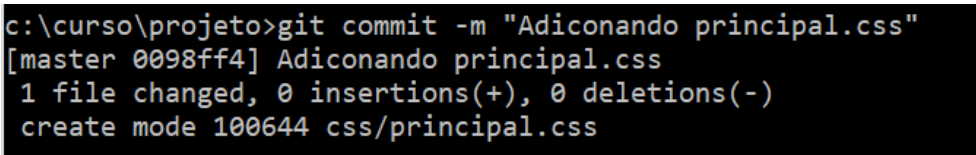
```
git status
```

Teremos:



```
c:\curso\projeto>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   css/principal.css
```

Figura 59- arquivo adicionado ao stage- rastreado



```
c:\curso\projeto>git commit -m "Adiconando principal.css"
[master 0098ff4] Adiconando principal.css
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 css/principal.css
```

Figura 60-Gravando alterações – comitando

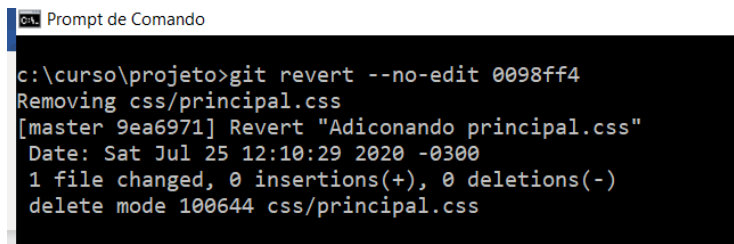
Pronto! Comitamos o arquivo `css/principal.css` ao diretório do projeto.

4.2.6.3 Desfazendo mudanças comitadas

E agora? Para desfazer mudanças já comitadas, podemos usar o comando:

```
git revert --no-edit 0098ff4
```

Na saída teremos:



```

c:\curso\projeto>git revert --no-edit 0098ff4
Removing css/principal.css
[master 9ea6971] Revert "Adiconando principal.css"
Date: Sat Jul 25 12:10:29 2020 -0300
1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 css/principal.css

```

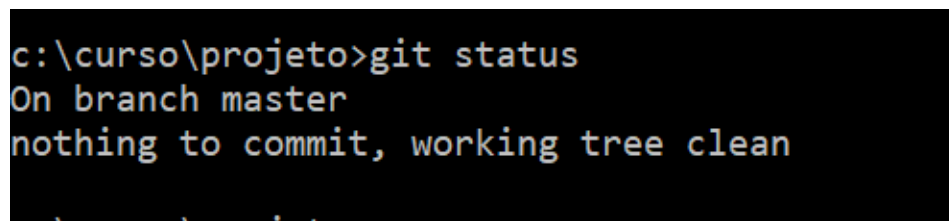
Figura 61- desfazendo alterações comitadas

Perceba que removemos o `css/principal.css`. Nesse comando, o código `0098ff4` representa o último `commit` realizado, conforme **figura 60**.

Após, execute o comando:

`git status`

Teremos:



```

c:\curso\projeto>git status
On branch master
nothing to commit, working tree clean

```

Figura 62- Nada para comitar

Bons estudos!

rogerio.pereira@ifto.edu.br

5 Referencias

ALMEIDA, Adriano ; MATSUI, Vivian. **Controlando Versões com Git e Github**. Casa do Código, São Paulo- Brasil.

CHACON E STRAUB. Pro git. 2ª edição, 2014. Disponível em: <https://git-scm.com/book/en/v2>. acesso em:14 de julho 2020.

FRAMEWORK.**Blog Relevo**. Disponível em: <https://blog.revelo.com.br/o-que-e-framework-exemplos-e-aplicacoes/> . Acesso em: 29, junho 2020.

GIT. **What is Git**. Disponível em: <https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F>. acesso em:14 de julho 2020.

HTML home. W3schools.com, Tutorial HTML . Disponível em: <https://www.w3schools.com/html/default.asp> . Acesso em: 29, junho 2020.

IDE. Wikipedia. Disponível em : https://pt.wikipedia.org/wiki/Ambiente_de_desenvolvimento_integrado. Acesso em: 29, junho 2020.

PÁGINA Web. **Wikipedia**. Disponível em: https://pt.wikipedia.org/wiki/P%C3%A1gina_web . Acesso em: 29, junho 2020.

SUBLIME Text 3. **Melhor Hospedagem de Sites**. Disponível em : <https://www.melhorhospedagemdesites.com/dicas-e-ferramentas/sublime-text-editor/> . Acesso em: 29, junho 2020.