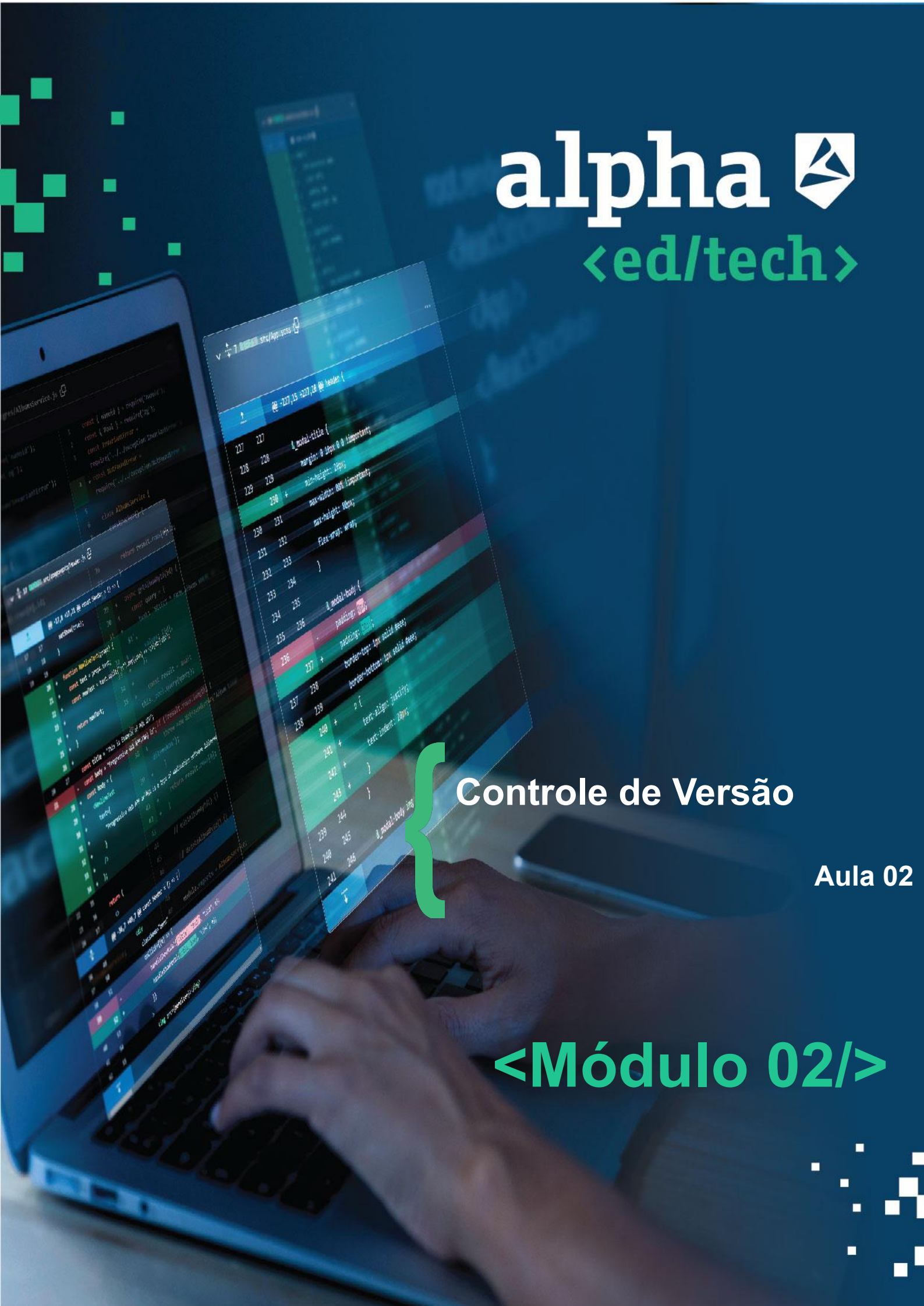
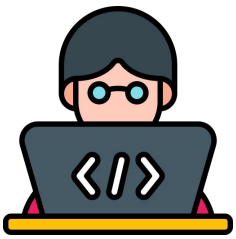

alpha
<ed/tech>
Controle de Versão
Aula 02
<Módulo 02/>

Visual Studio Code e GitLens

Introdução



Até agora, provavelmente você já cansou de ter que digitar no terminal toda pequena operação que deseja fazer.

Criar arquivos, editar com o `nano` ou outros editores, digitar comandos do Git...

Precisamos de uma maneira de simplificar esse processo, tornando-o mais visual e rápido.

Para isso, vamos introduzir um programa editor de texto chamado Visual Studio Code (VSCode), que oferece uma integração robusta com o Git.

O VSCode tem uma interface gráfica (janela) para editar qualquer arquivo sem precisar de outros programas, e também para controlar versões dos arquivos, identificar mudanças e realizar commits sem a necessidade de sair do ambiente de edição.

Ao todo, veremos os seguintes assuntos:

Instalação do Visual Studio Code
- Instalação no Windows
- Passos adicionais para WSL
- Instalação no Linux
- Instalação no macOS

Uso básico do VSCode
- Abrir uma pasta
- Explorador de Arquivos
- Criar um novo arquivo ou pasta
- Mover, renomear e deletar arquivos ou pastas
- Editar um arquivo
- Abrir o terminal

Funcionalidades Git no VSCode
- Introdução
- Ver mudanças do Index e Working Tree (git status, git diff)
- Adicionar mudanças ao Index (git add)
- Retirar mudanças do Index (git reset HEAD)
- Fazer um commit
- Remendar o último commit
- Ver versões antigas de um arquivo (git diff)

Extensão GitLens
- Introdução
- Instalação do GitLens
- Ver o histórico de commits (git log)
- Ver os detalhes de um commit

Instalação do Visual Studio Code

Instalação no Windows

Acesse o site oficial do Visual Studio Code e encontre a página de Downloads, ou acesse o link diretamente: https://code.visualstudio.com/Download

Faça o download do instalador para Windows:

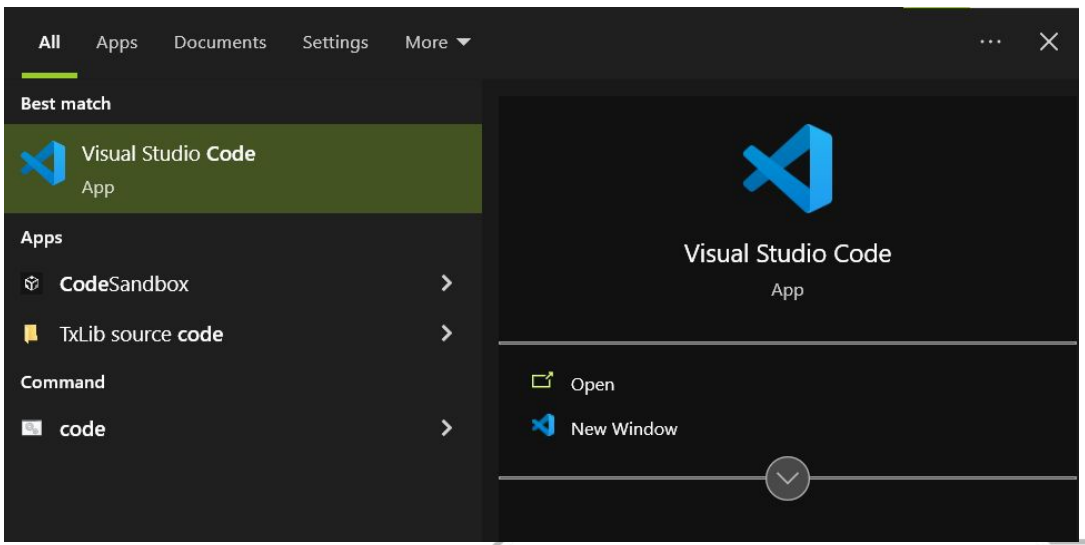
Siga o procedimento de instalação para Windows mesmo que você esteja usando WSL

Após finalizar o download, abra o instalador dando duplo-clique no arquivo.

Recomendamos marcar todas as opções de instalação seguintes:

- *"Criar um atalho na área de trabalho"*
- *"Adicione a ação"Abrir com Code" ao menu de contexto de arquivo do Windows Explorer"*
- *"Adicione a ação"Abrir com Code" ao menu de contexto de diretório do Windows Explorer"*
- *"Registre Code como um editor para tipos de arquivo suportados"*
- *"Adicione em PATH (disponível após reiniciar)"*

Uma vez instalado, você pode abrir o VSCode pelo ícone do programa na Área de Trabalho. Alternativamente, pode digitar `code` na barra de pesquisa:



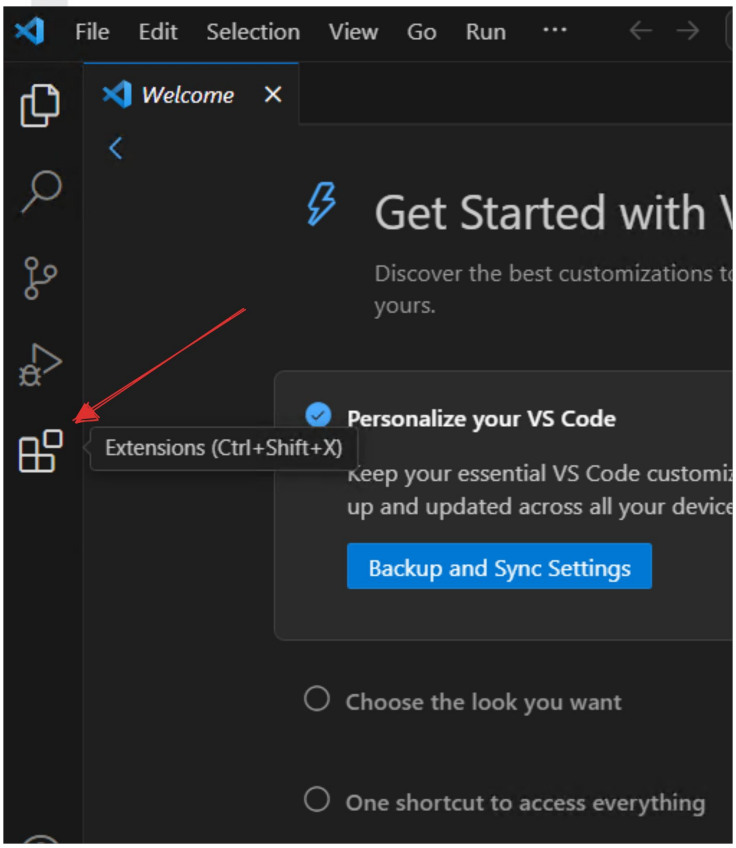
Instalação do Visual Studio Code

Passos adicionais para WSL

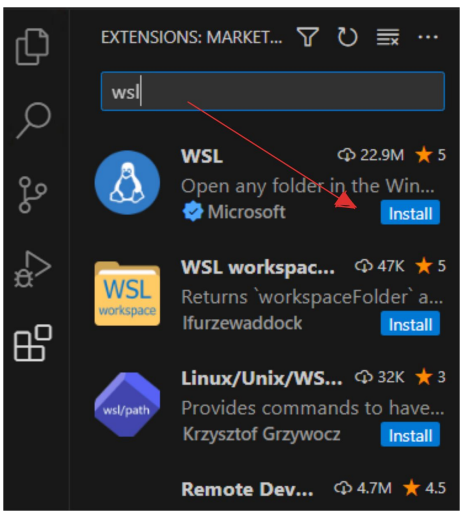
Se você está usando WSL no Windows, siga os passos de instalação do Windows primeiro.

Depois disso, como as pastas residentes no WSL são isoladas das pastas do Windows, é necessário configurar o VSCode para que ele possa acessar as pastas do WSL.

Para isso, abra o VSCode e clique no ícone de Extensões na barra esquerda, ou aperte `Ctrl+Shift+X` no teclado:



Então pesquise o termo `wsl` e clique em "Install"/"Instalar" no resultado da Microsoft:



Pronto, isso é suficiente para que o VSCode possa acessar as pastas do WSL.

Instalação do Visual Studio Code

Instalação no Linux

Acesse o site oficial do Visual Studio Code e encontre a página de Downloads, ou acesse o link diretamente: https://code.visualstudio.com/Download

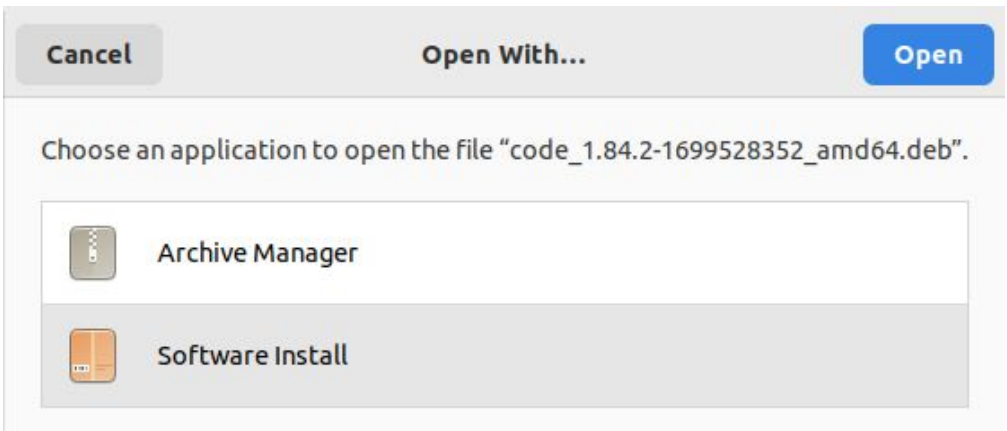
Você verá duas opções de instalação, uma para Debian e Ubuntu, outra para Red Hat, Fedora e SUSE:

Escolha aquela correspondente a seu sistema.

Se você está usando WSL no Windows, siga os passos de instalação do Windows, não do Linux !

Faça o download do instalador e o abra dando duplo-clique no arquivo.

É possível que o sistema pergunte como você deseja abrir o arquivo instalador. Por exemplo no Ubuntu:



Nesse caso, escolha "Software Install".

Uma vez instalado, o VSCode pode ser encontrado e iniciado por meio da lista de aplicativos do computador.

Instalação do Visual Studio Code

Instalação no macOS

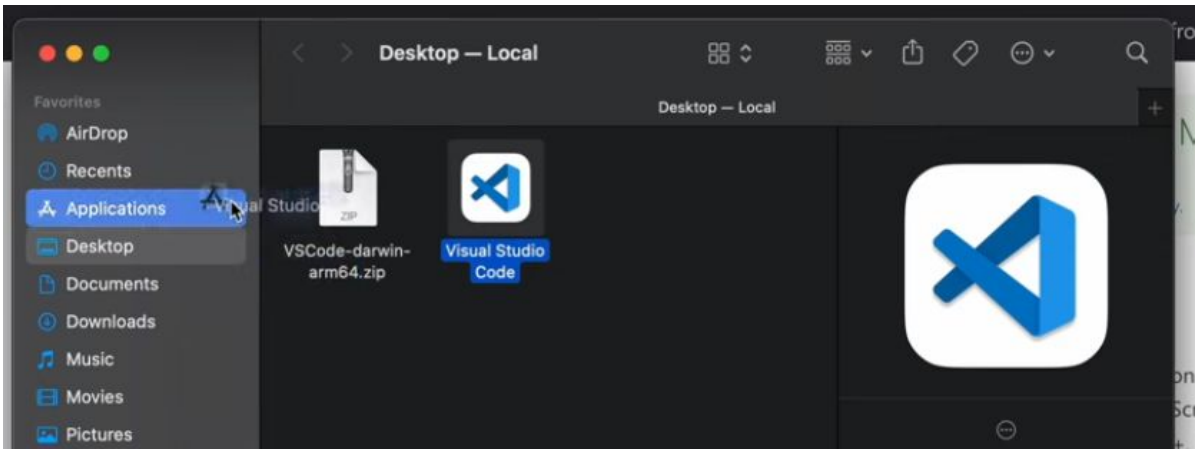
Acesse o site oficial do Visual Studio Code e encontre a página de Downloads, ou acesse o link diretamente: https://code.visualstudio.com/Download

Faça o download do instalador para macOS:

Uma vez finalizado o download, abra o instalador dando duplo-clique no arquivo.

Isso vai extrair o programa VSCode para a pasta em que o instalador se encontra.

Agora mova o programa para a pasta `Applications`:



O VSCode pode ser encontrado e iniciado por meio da lista de aplicativos do computador.

Uso básico do VSCode

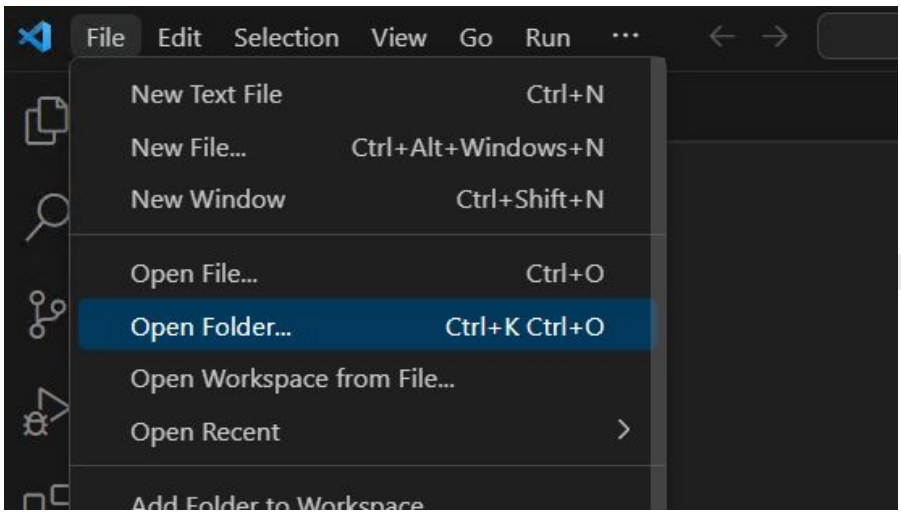
Introdução

Agora que você tem o VSCode instalado, veremos como usar suas funcionalidades básicas.

Abrir uma pasta

Para trabalharmos nos arquivos de um projeto com o VSCode, o primeiro passo é abrir nele a pasta do projeto.

No menu superior do VSCode, clique em "File"/"Arquivo", depois em "Open Folder"/"Abrir Pasta" e escolha a pasta do seu projeto:



Se seu projeto é uma pasta dentro do WSL no Windows, esse método não vai funcionar.

As pastas do WSL são isoladas das pastas do Windows, e não é fácil achá-las pelo Explorador de Arquivos.

A maneira mais fácil de abrir uma pasta do WSL é:

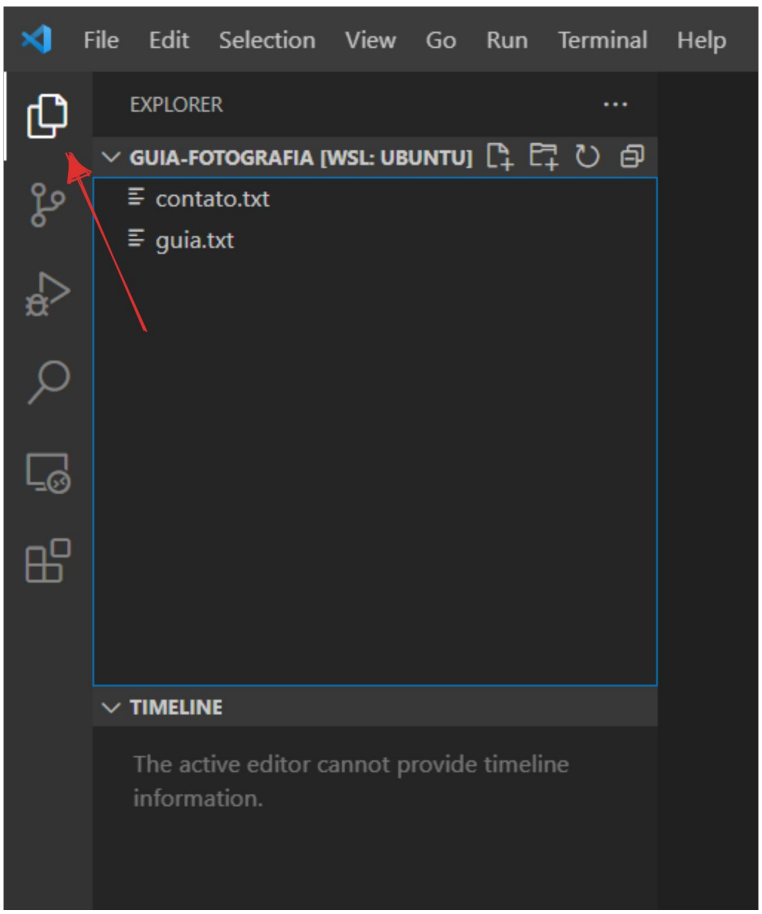
- Abrir um terminal do WSL
- Navegar até a pasta no terminal, com o comando `cd`
- Uma vez dentro da pasta, executar o comando `code .`, com um ponto no final como indicado

Isso vai abrir a pasta em uma nova janela do VSCode.

Uso básico do VSCode

Explorador de Arquivos

Uma vez com o VSCode aberto na pasta desejada, clique no ícone do Explorador de Arquivos na barra lateral esquerda:



Isso vai abrir o Explorador de Arquivos, que mostra a estrutura de diretórios e arquivos da pasta selecionada, que é `guia-fotografia` na imagem.

Os arquivos que você vê aqui são os mesmos que veria se abrisse o Explorador de Arquivos nativo do computador.

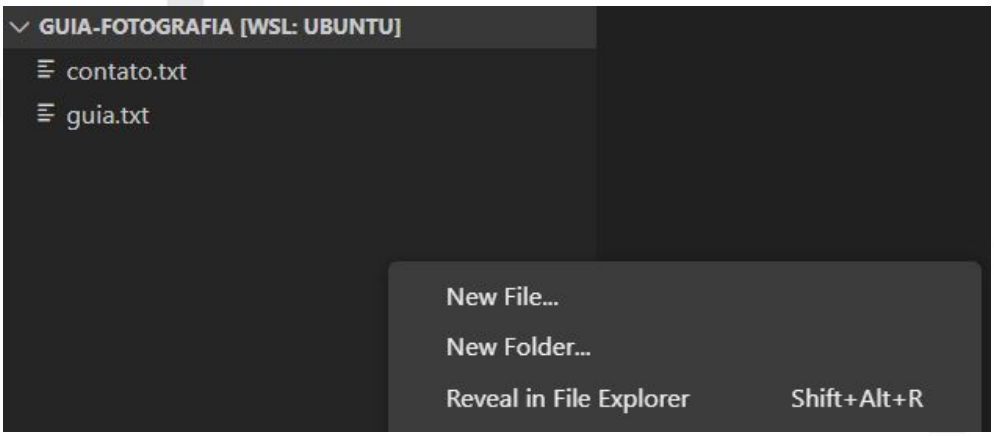
Em outras palavras, é a Working Tree do Git.

Uso básico do VSCode

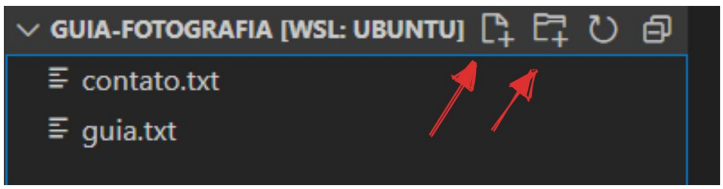
Criar um novo arquivo ou pasta

Há duas maneiras de criar um arquivo ou pasta.

1. Clique com o botão direito do mouse no Explorador de Arquivos, e então em "New File"/"Novo Arquivo" ou "New Folder"/"Nova Pasta":



2. Clique nos ícones de arquivo ou pasta que estão no topo do explorador de arquivos:



Nos dois casos, você precisará digitar um nome para o arquivo/pasta, e apertar `Enter` para efetivar a criação.

Mover, renomear e deletar arquivos ou pastas

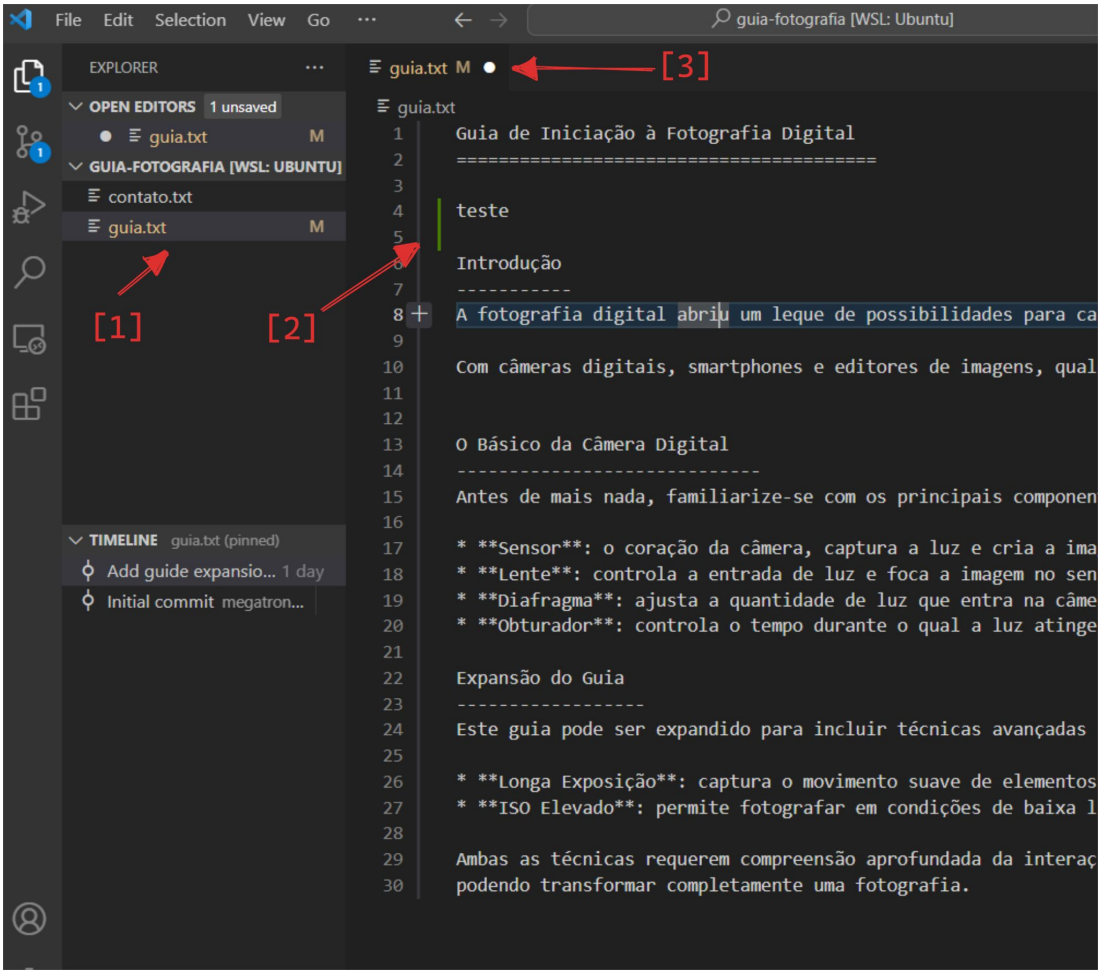
Arquivos e pastas no Explorador de Arquivos podem ser movidos clicando e arrastando com o botão esquerdo do mouse.

Se clicar com o botão direito do mouse, você poderá renomear ou deletar arquivos e pastas.

Uso básico do VSCode

Editar um arquivo

Acompanhe os pontos numerados abaixo:



1. Ao clicar num arquivo, ele é aberto do lado direito
2. Ao editar linhas do arquivo, elas ficam marcadas por uma barra colorida.

   Este é um exemplo de integração do VSCode ao Git: o VSCode sabe quais linhas foram alteradas porque ele faz internamente um `git diff` comparando seu arquivo com o último commit.
3. Enquanto o arquivo não for salvo, ficará uma bolinha ao lado do título.

   Para salvar, ou aperte `Ctrl+S` no teclado ou clique no menu "File"/"Arquivo", e então em "Save"/"Salvar"

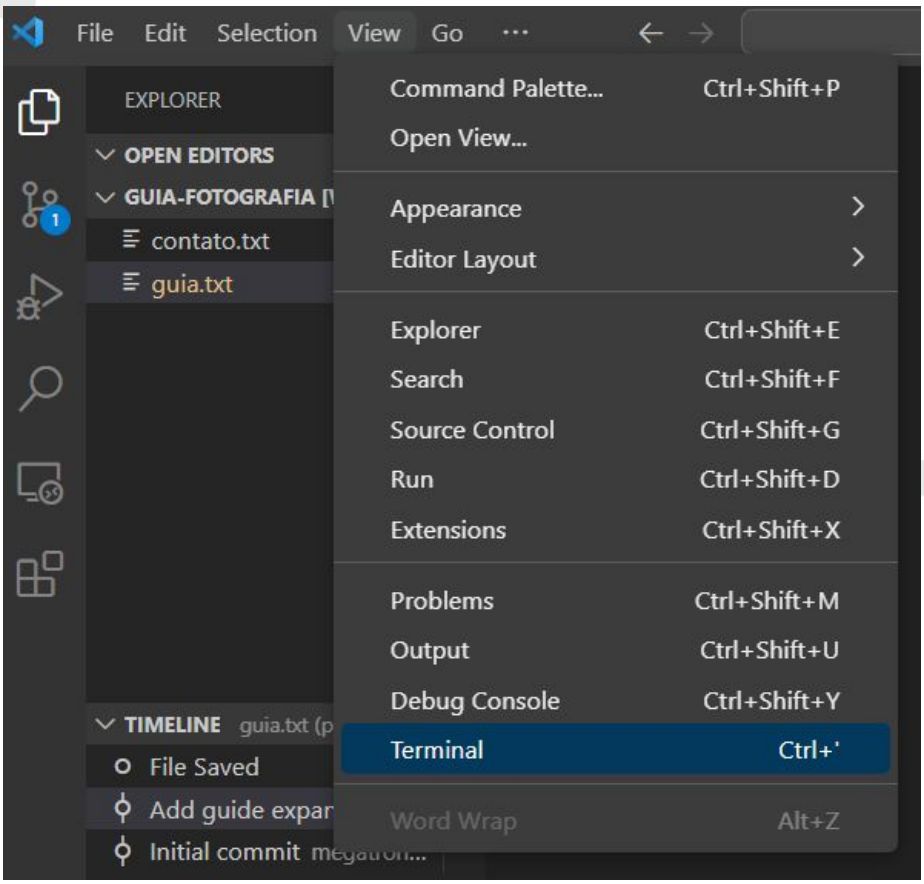
Outra funcionalidade útil é o "Desfazer" e "Refazer":

- Aperte `Ctrl+Z` no teclado para desfazer a modificação que você acabou de fazer ao arquivo.
- Aperte `Ctrl+Shift+Z` para refazer.

Uso básico do VSCode

Abrir o terminal

Por facilidade, o VSCode tem um terminal dentro do próprio editor.
Para abri-lo, clique no menu superior em "View"/"Visualizar", e então em "Terminal":



Funcionalidades Git no VSCode

Introdução

O VSCode permite fazer operações do Git diretamente dentro do editor.

A vantagem é que é mais fácil "clicar em botões" no editor do que digitar comandos no terminal.

O VSCode atua como um "mordomo" que faz o "trabalho pesado" de executar comandos do Git no terminal por você.

A seguir mostramos como realizar pelo VSCode vários dos comandos de Git que já sabemos fazer pelo terminal. Mas não passaremos por tudo que o VSCode consegue fazer.

Para os exemplos, vamos assumir que estamos trabalhando no projeto de um guia sobre fotografia digital, e que atualmente o projeto tem os seguintes commits, do mais recente ao mais antigo:

- Commit 2, com hash `5035`. Tem os arquivos:
  - `contato.txt` , que tem uma única linha informando seu email
  - `guia.txt`, com as seguintes seções de conteúdo:
    - Introdução
    - O Básico da Câmera Digital
    - Expansão do Guia (era "Modos de Disparo" no commit 1, você trocou)
- Commit 1, com hash `aef0`. Tem um único arquivo:
  - `guia.txt`, com as seguintes seções de conteúdo:
    - Introdução
    - O Básico da Câmera Digital
    - Modos de Disparo

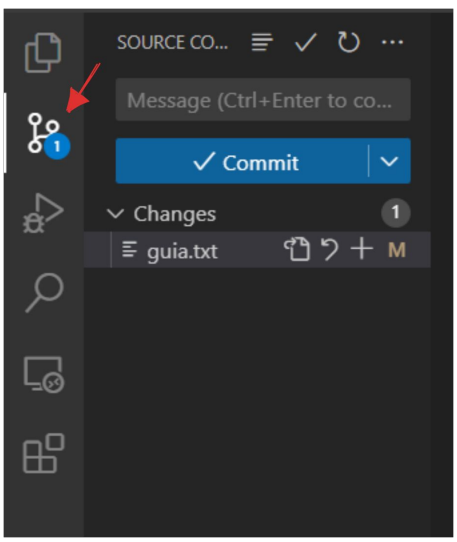
Depois do segundo commit, vamos assumir que adicionamos duas linhas ao `guia.txt` onde escrevemos a palavra "teste".

Mas não executamos nem `git add` nem `git commit`, ou seja, essas modificações estão em vermelho no `git status`.

Ver mudanças do Index e Working Tree (git status, git diff)

O equivalente do `git status` no VSCode é a aba Source Control (Controle de Fonte).

Para abri-la, clique no ícone do controle de versão na barra lateral esquerda, ou aperte `Ctrl+Shift+G`:



Funcionalidades Git no VSCode

A seção "Changes" corresponde ao que o `git status` mostraria em vermelho no terminal.

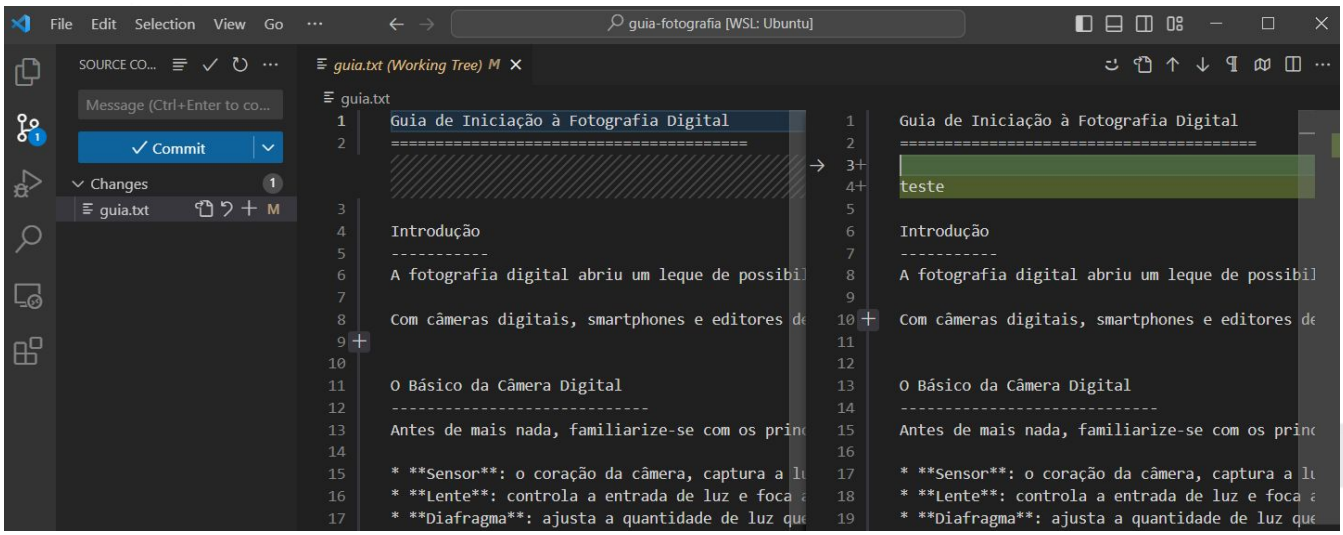
Ou seja, são as mudanças feitas na Working Tree que não foram adicionadas ao Index ainda.

Na imagem, aparece também a letra `M` ao lado do `guia.txt`. Isso significa que ele foi "Modificado".

Se fosse um arquivo novo, que (como já vimos) o `git status` chama de "Untracked file", apareceria no VSCode a letra `U` (de "Untracked").

E se fosse um arquivo que existia no Git e nós deletamos da Working Tree, apareceria no VSCode a letra `D` (de "Deleted").

Para ver exatamente quais mudanças foram feitas no `guia.txt`, clique no arquivo:



Do lado esquerdo é o arquivo no Index. Do lado direito é o mesmo arquivo na Working Tree.

Ou seja, nós adicionamos duas linhas ao arquivo.

Para comprovar que o VSCode está meramente representando as informações do Git que você já sabe acessar pelo terminal, vamos executar `git status` e `git diff` pelo terminal:

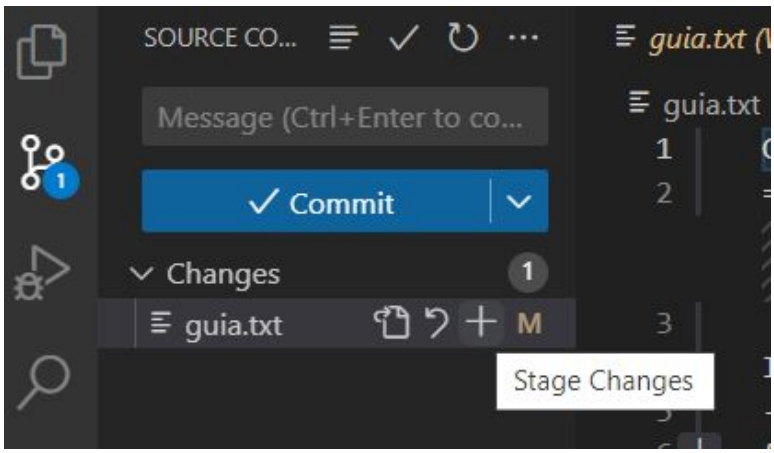
```
vitor@DESKTOP-QG84R3G:~/guia-fotografia$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   guia.txt

no changes added to commit (use "git add" and/or "git commit -a")
vitor@DESKTOP-QG84R3G:~/guia-fotografia$ git diff guia.txt
diff --git a/guia.txt b/guia.txt
index a44d801..318a5be 100644
--- a/guia.txt
+++ b/guia.txt
@@ -1,6 +1,8 @@
 Guia de Iniciação à Fotografia Digital
 ======================================

+teste^M
+^M
 Introdução
 ----------
 A fotografia digital abriu um leque de possibilidades para capturar mo
```

Confirmando que está tudo igual ao que aparece no VSCode.

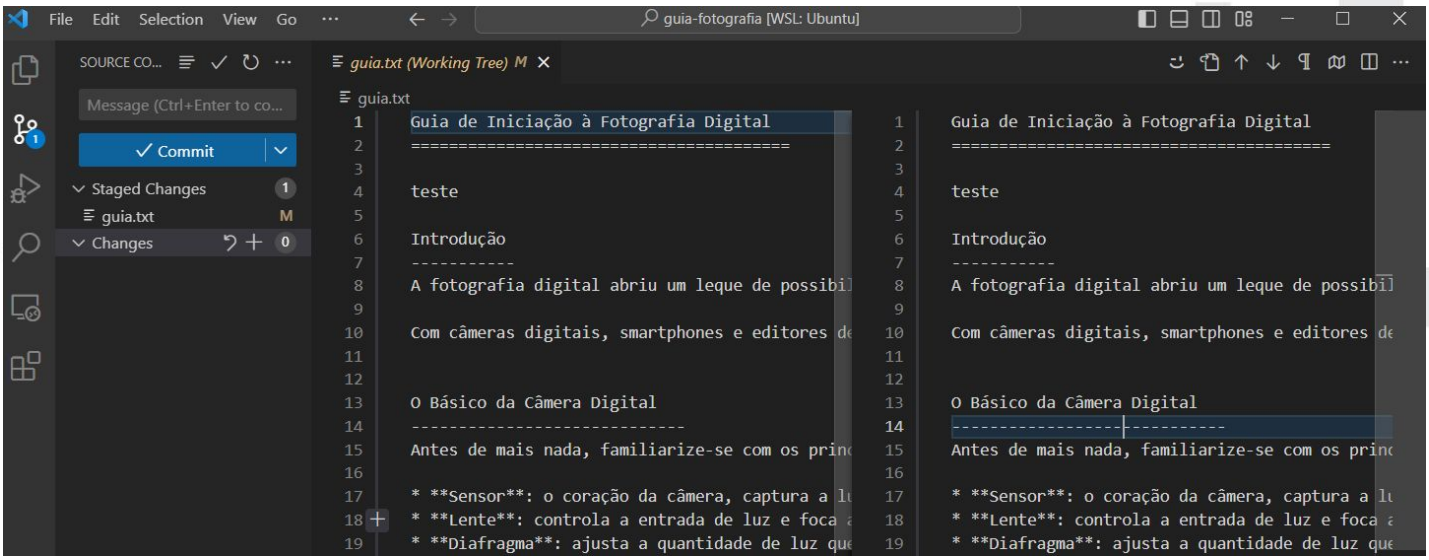
Funcionalidades Git no VSCode

Adicionar mudanças ao Index (git add)

Para adicionar as mudanças da Working Tree ao Index, clique no botão de + ao lado do arquivo na aba de Controle de Versão:



Agora a visão no VSCode muda:



Na aba de Controle de Versão, não se chama mais "Changes", mas sim "Staged Changes".

Isso significa que as mudanças foram adicionadas ao Index, ou seja, o `guia.txt` estaria em verde se você executasse `git status` agora pelo terminal.

E a visão de diff não mostra mais diferenças entre os arquivos, porque Index (lado esquerdo) e Working Tree (lado direito) são agora idênticos.

Funcionalidades Git no VSCode

Para comprovar que o VSCode fez meramente um `git add`, vamos usar o terminal:

```
vitor@DESKTOP-QG84R3G:~/guia-fotografia$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   guia.txt

vitor@DESKTOP-QG84R3G:~/guia-fotografia$ git diff guia.txt
vitor@DESKTOP-QG84R3G:~/guia-fotografia$
```

Isso corrobora o que vemos no VSCode: as mudanças foram adicionadas, e o diff está vazio.

Para as mudanças que já foram adicionadas ao Index, é possível fazer um diff entre o último commit e o Index.

Isso permite ver o que tem no Index que não tem no último commit, ou seja, o que fará parte do próximo commit caso você execute `git commit`.

Para fazer esse diff, basta clicar no arquivo dentro da seção "Staged Changes" do VSCode.

Isso abrirá um diff onde no lado esquerdo aparece o arquivo no último commit, e no lado direito aparece o arquivo no Index.

Você deve se lembrar que o `git status` no terminal pode mostrar o mesmo arquivo em vermelho e em verde ao mesmo tempo.

Já vimos que isso acontece quando algumas mudanças do arquivo foram adicionadas ao Index (então aparece em verde) e outras não (então aparece em vermelho).

No VSCode pode acontecer a mesma coisa: o mesmo arquivo pode estar tanto em "Staged Changes" quanto em "Changes".

- Se clicar no arquivo em "Changes", verá um diff entre o Index e a Working Tree.

  Ou seja, verá quais mudanças você ainda não adicionou ao Index.
- Se clicar no arquivo em "Staged Changes", verá um diff entre o último commit e o Index.

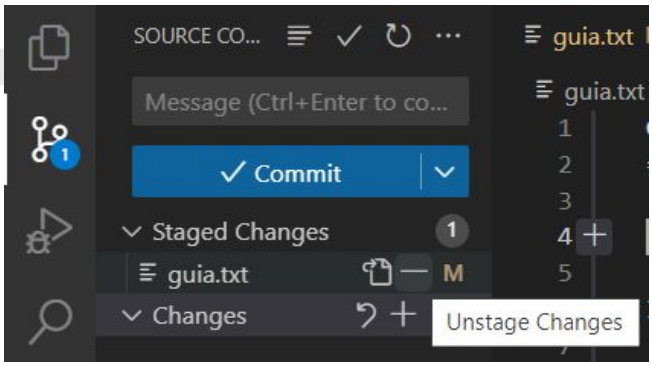
  Ou seja, verá quais mudanças estão marcadas para serem incluídas no próximo commit.

Funcionalidades Git no VSCode

Retirar mudanças do Index (git reset HEAD)

Já vimos que, se você dá `git add` em alguma modificação de um arquivo, mas depois se arrepende e quer retirá-la do Index, pode usar `git reset HEAD <nome do arquivo>`.

No VSCode é mais fácil, basta clicar no sinal de - que fica ao lado do arquivo em "Staged Changes":



Isso faz a mesma coisa que o comando de terminal.

Fazer um commit

Se você já clicou no + ou deu `git add` pelo terminal em todas as modificações que quer incluir no próximo commit, falta só o `git commit`.

No VSCode, basta digitar a mensagem na aba de Controle de Versão, e clicar no botão azul escrito "Commit"

Remendar o último commit

Já vimos que se você se esquece de dar `git add` em algumas modificações e depois faz `git commit`, ainda é possível incluir as modificações que faltaram.

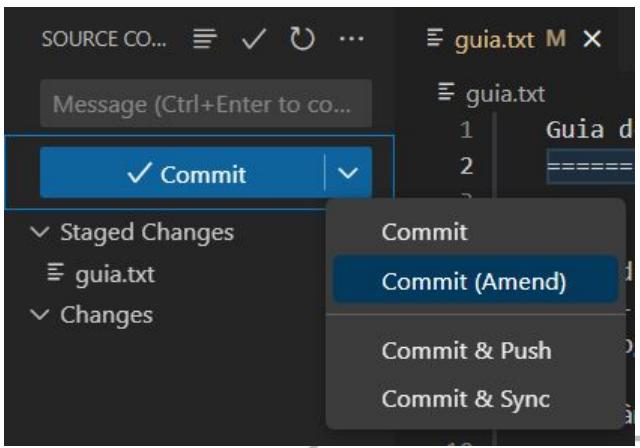
No terminal, você faria `git add` no que faltou e depois executaria `git commit --amend`.

Vimos que isso faz com que o último commit seja substituído por outro contendo as mesmas modificações de antes e adicionalmente as modificações em que você deu `git add`.

No VSCode, você pode fazer a mesma coisa.

Primeiro dê `git add` nas modificações que faltaram.

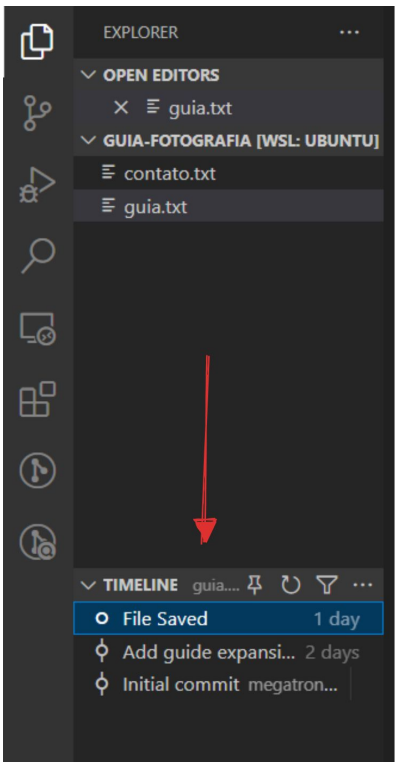
Depois, quando for fazer commit, clique na seta ao lado do botão "Commit", e escolha a opção "Commit (Amend)":



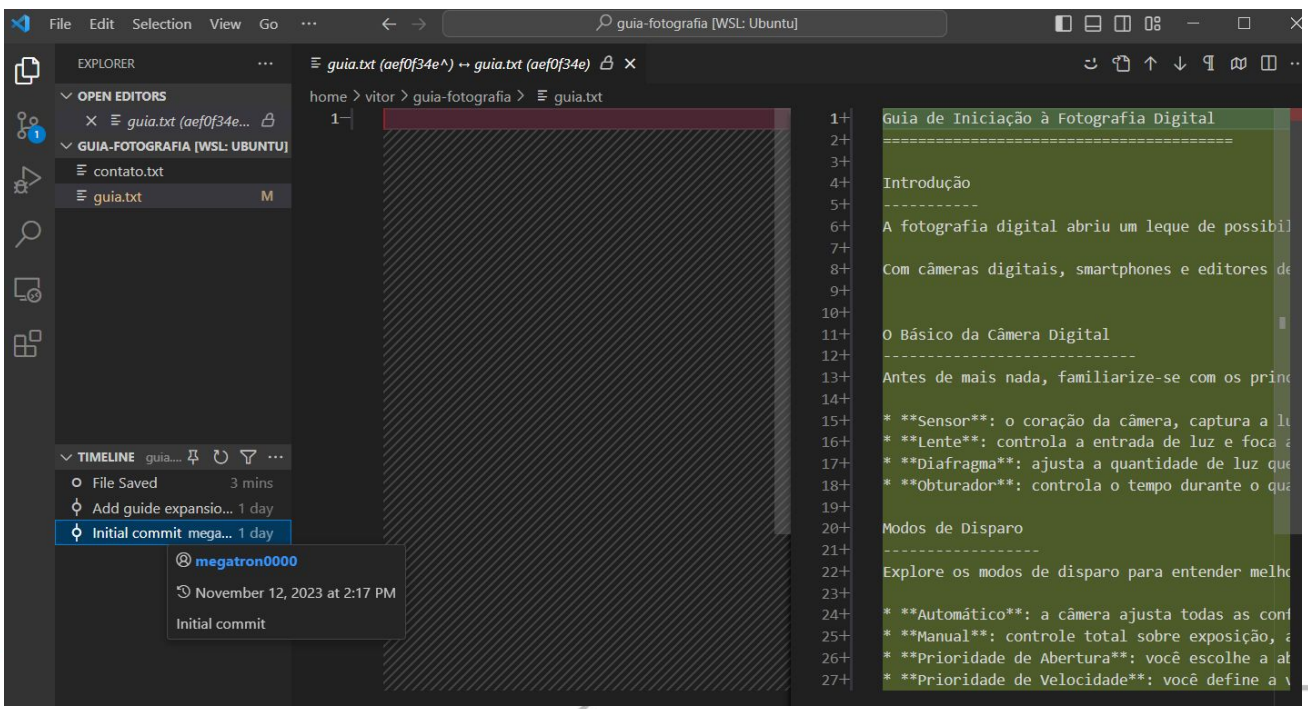
Funcionalidades Git no VSCode

Ver versões antigas de um arquivo (git diff)

Com um arquivo aberto, na aba de Explorador de Arquivos aparece a visão Timeline (Linha do Tempo).
Ela mostra as versões anteriores do arquivo. Por exemplo, abrindo o `guia.txt`, vemos o seguinte:



As bolinhas com um traço vertical são commits do Git.
E as bolinhas sem traço são versões que o próprio VSCode salva quando você edita e salva o arquivo.
Essas últimas não estão relacionadas com o Git, são um mecanismo de backup do próprio VSCode.
Ao clicar numa bolinha, o VSCode mostra um `git diff` para você.
Por exemplo, clicando no primeiro commit aparece o seguinte:

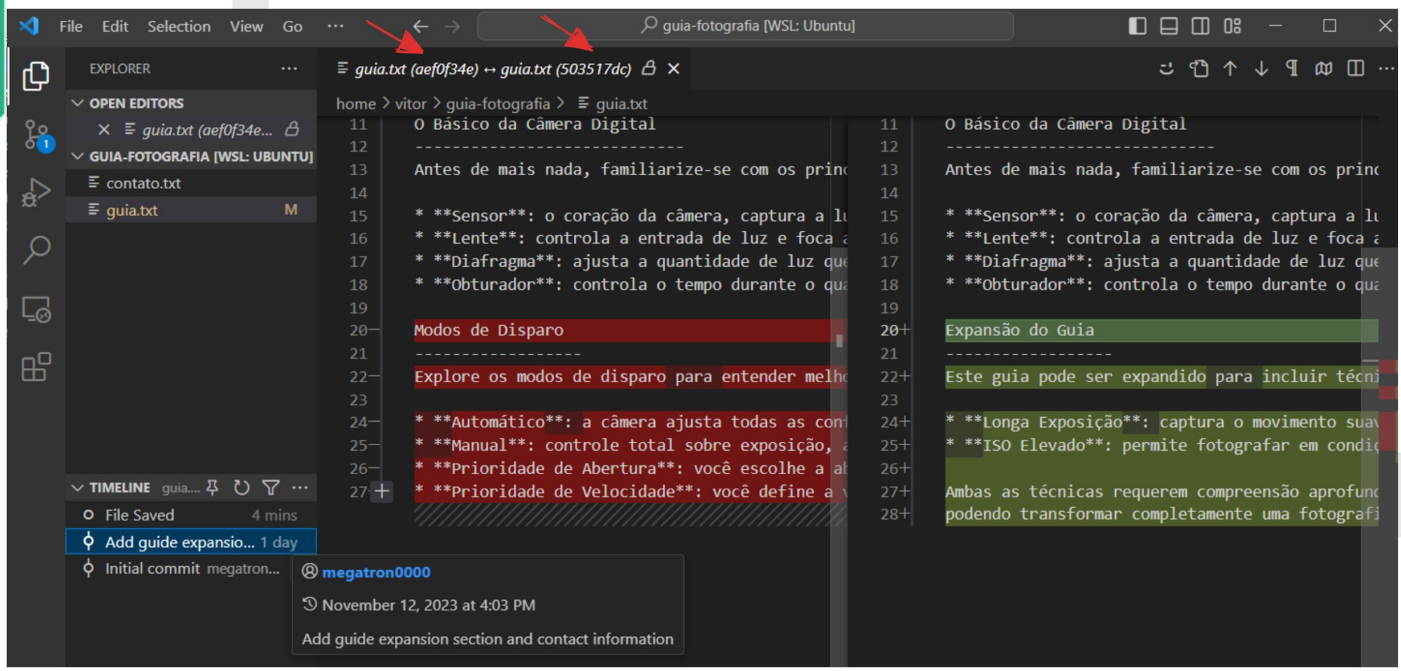


Funcionalidades Git no VSCode

Do lado esquerdo (vazio) é o arquivo como estava antes desse commit: o arquivo não existia, por isso está vazio.

Do lado direito é a versão do arquivo que foi salva nesse commit.

Como outro exemplo, ao clicarmos no segundo commit aparece o seguinte:



Ele está comparando agora o primeiro commit (lado esquerdo) contra o segundo (lado direito).

Olhando para isso, podemos ver que o arquivo tinha uma seção "Modos de Disparo" no primeiro commit, mas ela foi trocada para "Expansão do Guia" no segundo commit.

Note que no topo (setas vermelhas na imagem) aparecem os hashes dos dois commits sendo comparados.

Ao clicar num commit, o arquivo é comparado com o commit anterior ao clicado.

Em vez disso, pode ser que você queira comparar um commit com outro que não é o anterior.

Por exemplo, se há 10 commits, você pode querer comparar o 8º com o 3º.

Para isso, faça o seguinte:

1. Clique com o botão direito do mouse no 3º commit
2. Escolha a opção "Select for Compare"/"Selecionar para Comparação"
3. Clique com o botão direito do mouse no 8º commit
4. Escolha a opção "Compare with Selected"/"Comparar com o Selecionado"

Pronto ! O VSCode vai mostrar o diff.

Do lado esquerdo vai aparecer o arquivo como estava no 3º commit, e no lado direito como estava no 8º commit.

Extensão GitLens

Introdução

Extensões são módulos de software adicionais que podem ser instalados no VSCode para estender suas funcionalidades.

Imagine extensões como "apps" para o VSCode que acrescentam novos recursos e ferramentas.

Veremos aqui sobre uma delas, chamada GitLens, uma extensão que se integra ao Git.

Veremos como realizar certas operações de Git usando o GitLens, que na maioria não seriam possíveis no VSCode sem essa extensão.

Além das funcionalidades de Git que o VSCode já tem, o GitLens adiciona várias outras.

Por exemplo:

1. **Anotações de Linha:**

   O Git Lens permite ver quem fez as últimas alterações em uma linha específica do código.

   Isso é útil para entender o histórico e a autoria de um trecho de código.

2. **Blame Annotations:**

   Esta funcionalidade mostra informações detalhadas sobre as alterações feitas em um arquivo.

   Por exemplo o autor de cada alteração, a data e a mensagem de commit correspondente.

   Isso ajuda a rastrear a evolução do código ao longo do tempo.

3. **Exploração de Repositório:**

   Com o GitLens, você pode explorar facilmente o histórico de commits e tags do seu repositório Git diretamente no VSCode, sem a necessidade de usar comandos de terminal.

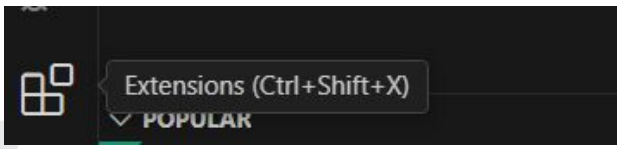
4. **Comparação de Alterações:**

   A extensão oferece uma maneira conveniente de comparar as diferenças entre commits, o que é essencial para revisar as alterações de código.

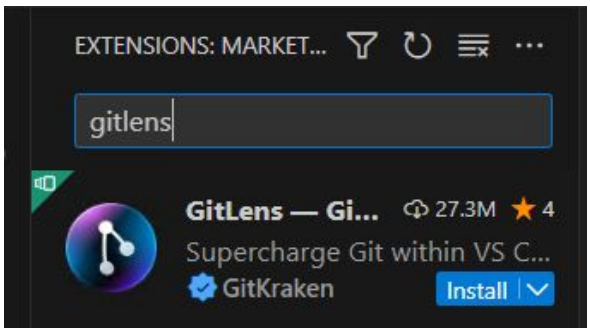
Extensão GitLens

Instalação do GitLens

Abra a aba de Extensões no VSCode clicando no ícone:



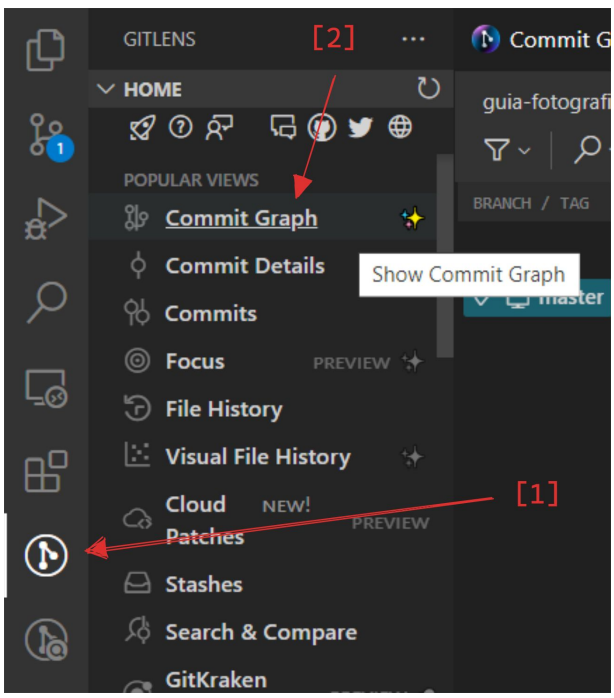
Pesquise por `gitlens` e clique em "Install"/"Instalar":



Ver o histórico de commits (git log)

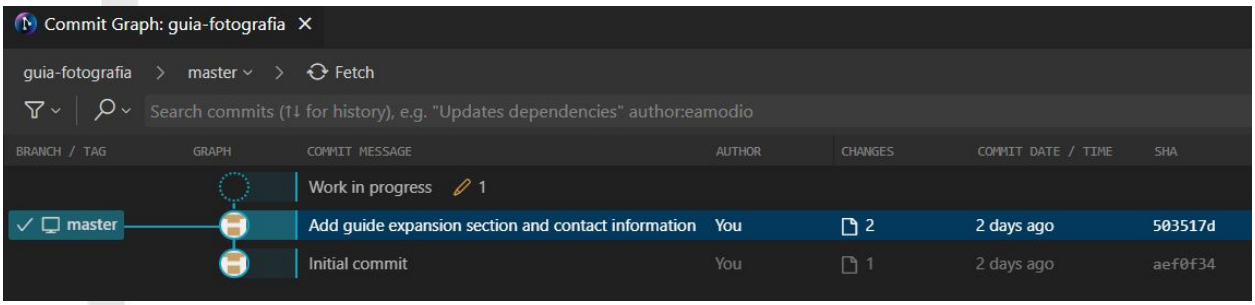
Uma funcionalidade que o VSCode não tem é a de explorar o histórico de commits.
O GitLens adiciona essa funcionalidade.
Para abrir a visão do histórico de Commits:

1. Abra a aba do GitLens clicando no ícone do menu lateral esquerdo
2. Clique na opção "Commit Graph"



Extensão GitLens

Do lado direito, vai aparecer a lista de commits:



Alguns elementos a observar:

- Os commits são apresentados do mais recente para o mais antigo, de cima para baixo.
- Existe uma bolinha tracejada com os dizeres "Work in progress" porque a Working Tree não está limpa, ela tem alguma(s) modificação(ões) desde o último commit.

  Se fizermos mais um commit, a bolinha tracejada vai "se tornar" o novo commit.
- A coluna "SHA" é o hash do commit.

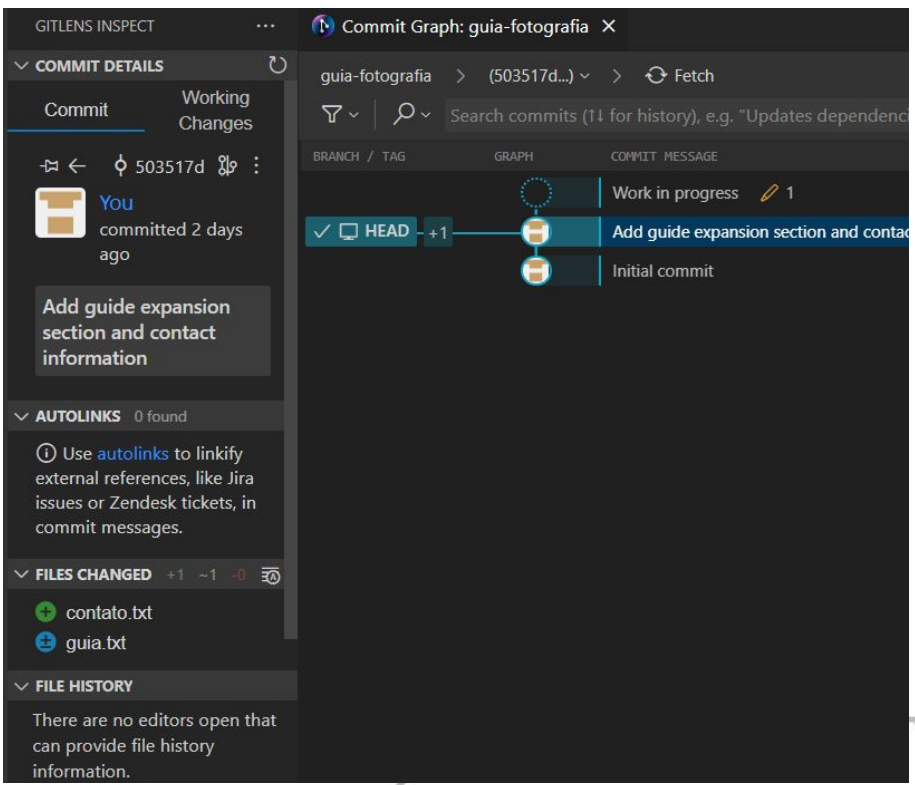
  O nome vem do algoritmo que o Git usa internamente para criar os hashes de commit, chamado "SHA-1"
- Não se preocupe por enquanto com a palavra "master" do lado esquerdo.

  Tecnicamente, ela é chamada de "branch" no Git, mas ainda veremos sobre isso em aulas futuras.

Ver os detalhes de um commit

Ainda na aba de histórico do GitLens, ao dar duplo-clique em cima de um dos commits, vai aparecer do lado esquerdo a tela de detalhes do commit.

Por exemplo, clicando no commit mais recente:

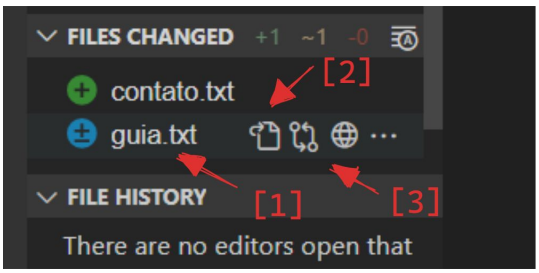


Extensão GitLens

Na seção "Files Changed" aparecem quais modificações foram salvas neste commit:

- O arquivo `contato.txt` foi criado.
  Isso é simbolizado pelo sinal `+` .
- O arquivo `guia.txt` foi modificado.
  Isso é simbolizado pelo sinal `±` .

Se você passar o mouse em um dos arquivos, terá 3 opções de visão do arquivo:



1. Clicar no nome do arquivo abrirá um diff entre o arquivo nesse commit e o mesmo arquivo no commit anterior
2. Clicar no ícone da esquerda abrirá o arquivo como ele estava naquele commit (não é um diff, é o arquivo em si)
3. Clicar no ícone da direita abrirá um diff entre o arquivo nesse commit e o mesmo arquivo na Working Tree