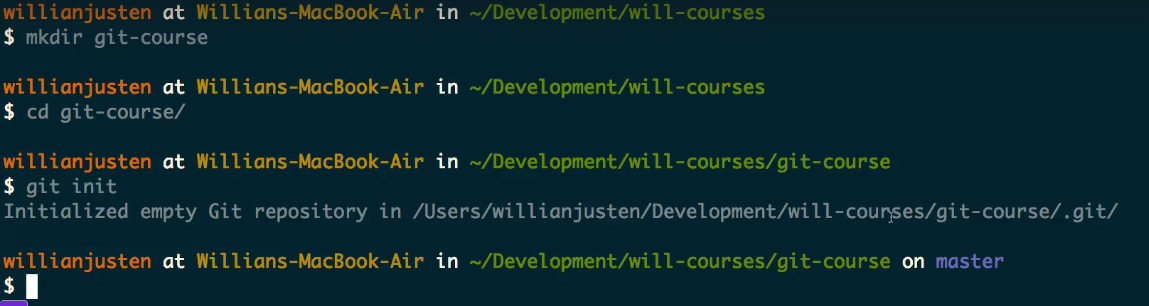
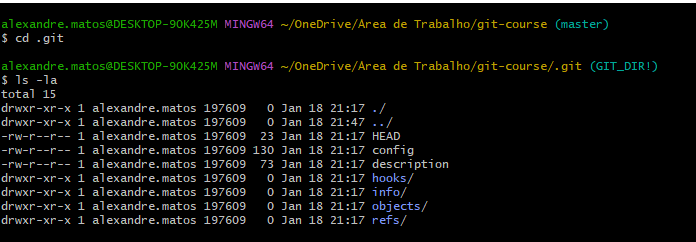
**GIT E GITHUB**

Abrindo o git na sua máquina ou o Terminal (Prompt de Comando) começamos criando uma pasta.



Após digitar **mkdir** e o nome da pasta que é **git-course** você dá **enter** e a pasta é criada;

**cd** é usado para acessar uma pasta, ex: cd git-course;



Na imagem que você enviou, você está dentro de um diretório chamado .git, que é o **diretório interno do Git** usado para armazenar todos os metadados do repositório. Esse diretório é automaticamente criado quando você inicializa um repositório com git init ou quando clona um repositório existente.

Aqui está uma explicação do que cada item listado significa:

1. **./** e **../**:
   * ./: Refere-se ao diretório atual (o próprio .git).
   * ../: Refere-se ao diretório pai (no caso, o diretório git-course).
2. **HEAD**:
   * Um arquivo que aponta para o commit ou branch atualmente "checado" no repositório. Ele determina onde você está trabalhando no momento.
3. **config**:
   * Contém as configurações específicas do repositório Git, como informações de usuário, branches remotos, etc.
4. **description**:
   * Um arquivo simples usado em repositórios "bare" (sem um diretório de trabalho) para descrever o repositório. Geralmente não é importante em repositórios locais.
5. **hooks/**:
   * Um diretório que contém scripts de **hooks do Git**, que permitem executar comandos personalizados em eventos específicos (como pre-commit ou post-merge).
6. **info/**:
   * Contém informações adicionais, como um arquivo exclude, que permite ignorar arquivos localmente (funciona de forma semelhante ao .gitignore, mas não é compartilhado com outros usuários).
7. **objects/**:
   * Um dos diretórios mais importantes. Contém todos os objetos do Git, como commits, árvores (trees), blobs (conteúdo dos arquivos) e tags. Tudo que você faz no Git (como adicionar, commitar, etc.) é registrado aqui.
8. **refs/**:
   * Contém referências para branches e tags no repositório. Por exemplo, as referências para main, develop ou outras branches criadas.

**Contexto Geral**

O que você está vendo é a estrutura interna de um repositório Git. Geralmente, você não precisa acessar o .git diretamente, pois o Git gerencia automaticamente esses arquivos e diretórios. No entanto, isso é útil para entender como o Git armazena informações e opera nos bastidores.



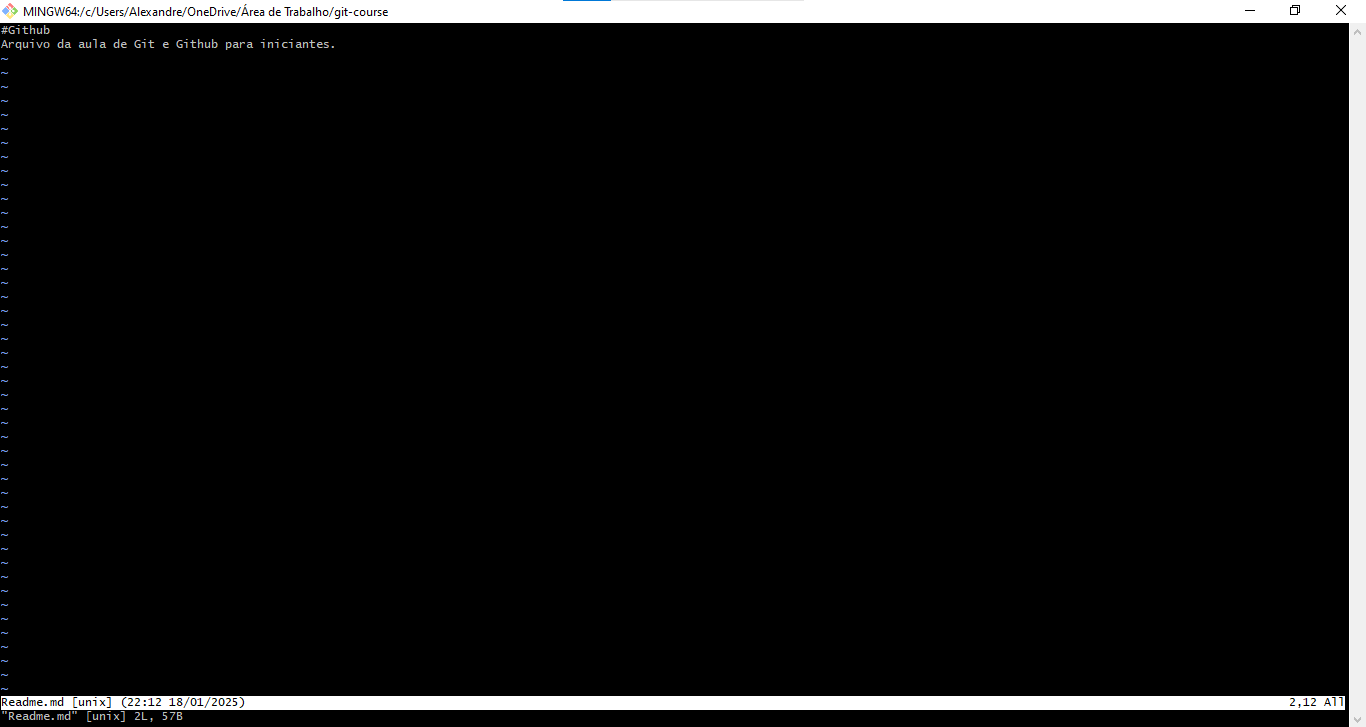
**ls**: Lista os arquivos e diretórios do diretório atual.

**-l**: Mostra os detalhes em formato de lista longa, incluindo:

* Permissões dos arquivos/diretórios.
* Número de links.
* Proprietário.
* Grupo.
* Tamanho.
* Data e hora da última modificação.
* Nome do arquivo/diretório.

**-a**: Mostra todos os arquivos, incluindo os ocultos (arquivos ou diretórios que começam com um ponto, como .git).

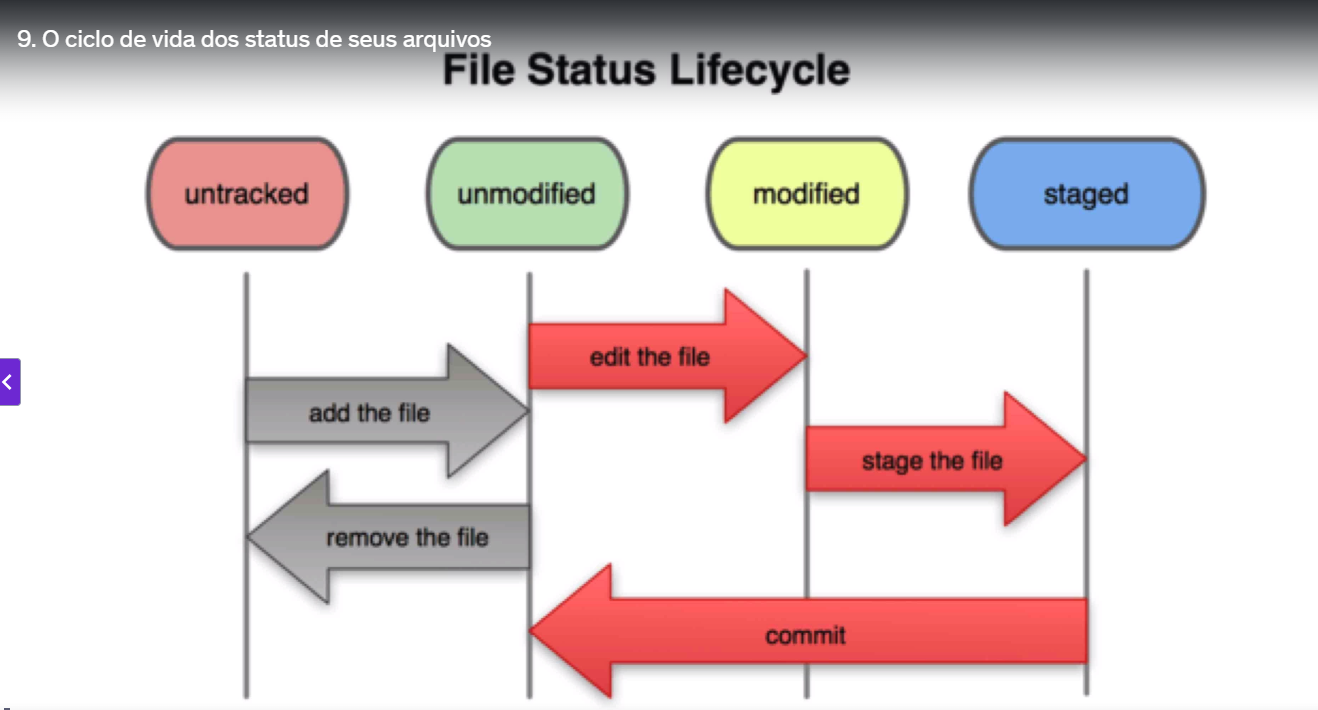




Ao você digitar: **vi** e o nome do arquivo **Readme.md,** você entrará no modo de edição do arquivo;

Depois seleciona a tecla **i** para inserir (insert);

Ao terminar de fazer a inserção só pertar **Esc** e em seguida clicar em **:** e depois em **wq** e **Enter** e o arquivo será salvo com a modificação.



O ciclo de vida dos status dos arquivos no Git, representado na imagem, descreve as diferentes etapas pelas quais um arquivo pode passar dentro de um repositório Git. Aqui está o significado de cada status e a transição entre eles:

### 1. ****Untracked**** (Vermelho - Não rastreado):

* Representa arquivos que existem no diretório de trabalho, mas que o Git ainda não está rastreando.
* Esses arquivos não fazem parte do controle de versão.

#### Como mudar de Untracked para Staged:

* Use o comando git add <arquivo> para adicionar o arquivo ao índice (área de stage).

### 2. ****Unmodified**** (Verde - Não modificado):

* São arquivos que já estão sendo rastreados pelo Git e que **não foram modificados** desde o último commit.
* Eles estão no estado "limpo".

#### Como mudar de Unmodified para Modified:

* Edite o arquivo.

### 3. ****Modified**** (Amarelo - Modificado):

* Representa arquivos que foram **modificados** desde o último commit.
* O Git detecta as alterações, mas elas ainda **não foram preparadas para commit**.

#### Como mudar de Modified para Staged:

* Use o comando git add <arquivo> para mover o arquivo modificado para a área de stage.

### 4. ****Staged**** (Azul - Preparado):

* Arquivos que estão na **área de stage** e prontos para serem incluídos no próximo commit.
* O estágio serve como uma etapa intermediária antes de salvar as alterações no histórico do repositório.

#### Como mudar de Staged para Committed:

* Use o comando git commit -m "<mensagem>" para registrar as alterações no repositório.

### Transições no Ciclo de Vida:

1. **Untracked → Staged**: Adicione o arquivo com git add.
2. **Unmodified → Modified**: Edite o arquivo.
3. **Modified → Staged**: Use git add.
4. **Staged → Committed**: Use git commit.

### Resumo:

* O ciclo reflete como o Git gerencia arquivos para garantir que apenas as alterações necessárias sejam rastreadas e incluídas no histórico do repositório.
* Este fluxo ajuda a organizar as alterações antes de registrá-las com um commit. Isso é útil para manter o controle e evitar erros.

Se precisar de mais explicações ou exemplos práticos, é só pedir! 😊