Perguntas

1. **Por que o Git é considerado um sistema de controle de versão distribuído?** 💡

   Resposta: Porque, diferente dos sistemas antigos onde tudo ficava num servidor central, no Git todo mundo que baixa o projeto tem uma cópia completa do histórico no próprio PC. Se o servidor explodir, a gente não perde nada porque todo mundo tem um backup local.

2. **Qual a diferença entre working directory, staging area e repository?** 💡

   Resposta: O *working directory* é a pasta normal onde tô mexendo nos arquivos agora. O *staging area* é tipo uma "sala de espera" onde eu separo só o que eu quero salvar no próximo pacote. O *repository* é o banco de dados definitivo onde essas versões (commits) ficam guardadas para sempre.

3. **Para que serve o comando git clone?** 💡

   Resposta: Serve pra baixar um projeto que já existe (tipo do GitHub) pro meu computador. Ele não baixa só os arquivos, baixa todo o histórico de mudanças junto.

4. **Onde estão implementados fisicamente working directory, staging area e repository?** 💡

   Resposta: O *working directory* é a pasta visível do projeto. O *staging area* e o *repository* ficam escondidos dentro daquela pasta oculta `.git` que aparece quando a gente inicia o projeto.

5. **Quais os estados de um arquivo no repositório do git?** 💡

   Resposta: Pelo que entendi, são quatro principais: Untracked (o Git não conhece), Unmodified (tá salvo e sem mudanças), Modified (eu mexi, mas não salvei ainda) e Staged (tá preparado pra ser salvo).

6. **Explique as possíveis transições de estado de um arquivo no repositório do git?** 💡

   Resposta: Quando crio o arquivo, ele é *Untracked*. Dou `git add`, ele vira *Staged*. Dou `git commit`, ele vira *Unmodified* (tá salvo!). Se editado de novo, ele fica *Modified*. Aí tenho que dar `add` de novo pra voltar pra *Staged*.

Etapa 1 – Criar o repositório

Pergunta: Qual foi a mensagem exibida após o comando git init e o que ela significa na prática? 💡

Resposta: Ele disse `Initialized empty Git repository in C:/Users/anart/OneDrive/Área de Trabalho/aula-git/.git/`. Na prática, ele criou a pasta oculta `.git` e começou a vigiar minha pasta `aula-git`.

1. **Qual o estado do arquivo antes e depois do git add?** 💡

   Resposta: Antes estava *Untracked* (vermelho no `git status`), depois ficou *Staged* (verde).

2. **O que significa o estado untracked e tracked?** 💡

   Resposta: *Untracked* é quando o Git ignora o arquivo. *Tracked* é quando o Git já "conhece" o arquivo e vigia qualquer mudança nele.

3. **Qual o objetivo do git commit?** 💡

   Resposta: É fechar o pacote e salvar uma versão de verdade no histórico. É como tirar uma foto de como o projeto está naquele momento.

4. **Qual o estado do arquivo após o git commit?** 💡

   Resposta: Ele fica *Unmodified* (limpo), porque a versão salva agora é igual à que está na pasta.

1. **O que o comando git diff mostra?** 💡

   Resposta: Mostra as linhas exatas que eu mudei no arquivo desde o último commit (o que eu adicionei ou apaguei) mas que ainda não dei `git add`.

2. **Qual commit está atualmente apontado por HEAD?** 💡

   Resposta: O HEAD tá apontando pro último que eu fiz, que tem a mensagem "Primeiro commit" (a menos que eu já tenha feito outro).

1. **Como verificar em qual branch você está?** 💡

   Resposta: Rodando `git branch`. A que tiver um asterisco `*` e estiver colorida é a atual. O `git status` também fala.

2. **O que acontece se você rodar git merge nova-feature estando na branch principal?** 💡

Resposta: Ele vai pegar as mudanças que eu fiz lá na branch `nova-feature` e trazer pra branch principal, juntando os dois históricos.

1. **O que significa o -u no comando git push -u origin main? 💡**

   Resposta: O `-u` serve pra "casar" a minha branch `main` local com a `main` lá do GitHub. Assim, nas próximas vezes eu só digito `git push` e ele já sabe pra onde mandar.

2. **Como verificar os remotes configurados no repositório? 💡**

   Resposta: Usando o comando `git remote -v`.

Qual etapa foi mais difícil? 💡

Resposta: Acho que lembrar de sempre fazer o `git add` antes do `git commit`. Às vezes eu esqueço e tento commitar direto, mas ele não vai. Entender o tal do "staging area" demorou um pouquinho pra cair a ficha.

Como o Git ajuda na colaboração? 💡

Resposta: Cada um pode trabalhar na sua própria branch sem quebrar o código principal. Depois a gente usa o merge pra juntar o trabalho de todo mundo de forma organizada.

Que diferença faz ter um repositório remoto? 💡

Resposta: É a segurança de ter um backup na nuvem (se meu PC quebrar, não perco o projeto) e é o ponto de encontro pra equipe toda poder baixar e enviar as atualizações.