**Perguntas – Parte 1**

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

Porque cada desenvolvedor mantém uma **cópia completa do repositório**, incluindo todo o seu histórico. Ou seja, não depende de um servidor central para operar: é possível fazer commits, criar branches e ver histórico mesmo offline.

---

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

- **Working directory:** é onde os arquivos estão no seu projeto local, editáveis.
- **Staging area:** é uma área temporária onde colocamos as mudanças que queremos versionar usando git add.
- **Repository (.git):** é onde ficam gravados os commits e todo o histórico do projeto.

---

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

Ele cria uma **cópia completa** de um repositório remoto na máquina local, incluindo arquivos, branches e histórico.

---

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

- **Working directory:** na pasta do projeto, visível ao usuário.
- **Staging area:** dentro do diretório .git, em um arquivo chamado *index*.
- **Repository:** também no diretório oculto .git, onde ficam os objetos e metadados.

---

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

- **Untracked**
- **Tracked**, que pode ser:
    - Modified
    - Staged
    - Committed

---

**6. Explique as possíveis transições de estado de um arquivo.**

- **Untracked → Staged:** quando damos git add pela primeira vez.
- **Staged → Committed:** quando fazemos git commit.
- **Committed → Modified:** quando editamos o arquivo.
- **Modified → Staged:** quando damos git add novamente após modificar.

---

**Perguntas – Parte 2**

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

- **Antes:** normalmente está como *modified* ou *untracked*.
- **Depois:** passa a ficar em *staged*.

---

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

- **Untracked:** arquivo existe no diretório, mas ainda não faz parte do versionamento.
- **Tracked:** o Git já controla o arquivo; ele pode estar committed, modified ou staged.

---

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

Registrar oficialmente as alterações da staging area no repositório, criando um novo ponto no histórico.

---

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

Ele passa a estar em estado **committed**, ou seja, salvo no histórico.

---

**Perguntas – Parte 3**

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

Mostra **diferenças entre versões de arquivos**, como o que foi alterado entre working directory e staging, ou entre commits.

---

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

O HEAD aponta para o **commit atual da branch ativa** (normalmente o último commit da branch onde estamos trabalhando).

---

**Perguntas – Parte 4**

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

Com o comando:

git branch

A branch atual aparece com um asterisco (*).

---

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

A branch principal tentará integrar (mesclar) as alterações da branch **nova-feature**. Se tudo estiver compatível, será criado um commit de merge; se houver conflitos, será necessário resolvê-los.

---

**Encerramento e Discussão**

**Qual etapa foi mais difícil?**

Normalmente, entender os estados dos arquivos (untracked, modified, staged, committed) e como eles mudam é o que mais confunde no início.

---

**Como o Git ajuda na colaboração?**

Ele permite que várias pessoas trabalhem no mesmo projeto, cada uma na sua própria cópia, sem sobrescrever o trabalho dos outros. O Git controla versões, identifica conflitos e registra quem fez o quê.

---

**Que diferença faz ter um repositório remoto?**

Um repositório remoto permite **compartilhar o código**, fazer backups e sincronizar o trabalho entre diferentes máquinas e equipes. Sem ele, tudo ficaria apenas no computador local.