Prática: Git e GitHub

Nome: Renato Costa Giuliano (32329307)

1. Introdução e Conceitos

1. Por que o Git é considerado um sistema de controle de versão distribuído? 💡 Resposta: Porque ele permite que cada desenvolvedor tenha uma cópia completa do repositório em seu computador local.

2. Qual a diferença entre working directory, staging area e repository? 💡 Resposta: Working directory é o diretório onde o trabalho é feito, onde podemos trabalhar com os arquivos (mover, excluir, editar, etc). Staging area é um arquivo que armazena quais alterações do working directory serão incluídas na próxima snapshot. Repository é o histórico do projeto, onde são armazenados todos os commits e branches, os arquivos são considerados seguros e registrados, o conteúdo da staging area é movido para repository quando se finaliza um ciclo de trabalho.

3. Para que serve o comando git clone ? 💡 Resposta: O git clone serve para criar uma cópia localmente, de um repositório já existente. O comando faz que a cópia seja criada no seu computador com todo o histórico de commits, branches, tags e etc.

4. Onde estão implementados fisicamente working directory, staging area e repository ? 💡 Resposta: Working directory é a própria pasta ou diretório do projeto, contém a versão real dos arquivos, é a cópia atual do projeto. Directory está na sub pasta oculta chamada *.git/* que fica na raiz do working directory. Staging area é o arquivo *index* que fica dentro do diretório *.git/*

5. Quais os estados de um arquivo no repositório do git ? 💡 Resposta: São 4 estados: Untracked, unmodified, modified e staged

6. Explique as possíveis transições de estado de um arquivo no repositório do git ? 💡 Resposta: Quando começamos criando um arquivo, ele é untracked, ainda não foi reconhecido pelo git. Então é inserido o comando *git add,* o git agora reconhece o arquivo e ele está no estado staged.
Posteriormente devemos usar o comando *git commit,* nessa parte o snapshot staged é salvo no repository, o arquivo passa de staged para unmodified.

Ainda é possível alterar novamente o arquivo, se isso acontecer, o arquivo muda o estado de unmodified para modified, pois o Git detecta que a versão no working directory é diferente da versão salva no último commit. Então após usar o *git commit* ele voltaria para unmodified.

2. Prática com Git Local

Etapa 1 – Criar o repositório

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

Initialized empty Git repository in D:/GIT/Git/aula-git/.git/

Significa que o comando foi bem sucedido, foi criado um repositório Git local em um diretório existente. Ele foi inicializado como um repositório vazio “empty”, ou seja, sem nenhum commit.

Etapa 2 – Adicionar arquivo e fazer commit

1. Qual o estado do arquivo antes e depois do git add ? 💡
Resposta: O comando *git add* moveu as alterações do working

directory para o staging area, então o arquivo passou de untracked para staged.

2. O que significa o estado untracked e tracked ? 💡 Resposta: Untracked é um arquivo que o Git nunca salvou, ele não conhece o arquivo, ainda não foi usado o comando *git add.*
Tracked é um arquivo que o Git já salvou pelo menos alguma vez, eles são divididos em unmodified, modified e staged.

3. Qual o objetivo do git commit ? 💡 Resposta: O *git commit* tem como função salvar a snapshot do que foi preparado na staging area.

4. Qual o estado do arquivo após o git commit ? 💡 Resposta: Antes do comando *git commit* o arquivo estava no estado staged, após o comando, o arquivo fica no estado unmodified.

Etapa 3 – Histórico e alterações

1. O que o comando git diff mostra? 💡 Resposta: O comando *git diff* mostra o que foi alterado no arquivo, no caso dessa atividade o comando mostra “+Nova linha” como saída, pois acrescentamos coisas no arquivo. Se tivéssemos retirado alguma coisa do arquivo, a saída mostraria “-” e o que foi removido.

2. Qual commit está atualmente apontado por HEAD? 💡 Resposta: HEAD está atualmente apontando para a branch master, e a branch master aponta para o commit “primeiro commit”

Etapa 4 – Trabalhando com Branches

1. Como verificar em qual branch você está? 💡 Resposta: Podemos verificar a branch usando o comando *git branch*

2. O que acontece se você rodar git merge nova-feature estando na branch principal? 💡 **Resposta:** O git faz um fast-forward, o ponteiro é movido da branch principal, para a branch nova-feature.

3. Conectando ao GitHub

1. O que significa o -u no comando git push -u origin main ? 💡
Resposta: O -u tem a função de vincular o branch local com o branch remoto, ele define o branch remoto como upstream padrão.

2. Como verificar os remotes configurados no repositório? 💡
Resposta: Usando o comando *git remote*

4. Encerramento e Discussão

Qual etapa foi mais difícil? A primeira etapa, pois era necessário ter a concepção de várias ideias e conceitos diferentes para conseguir avançar na prática.

Como o Git ajuda na colaboração? Suas funcionalidades permitem que os desenvolvedores consigam trabalhar simultâneamente, aumentando a agilidade e a confiabilidade do processo.

Que diferença faz ter um repositório remoto? Ter um repositório remoto ajuda no processo de desenvolvimento de várias formas como: Trabalho em equipe, backup e segurança de dados, distribuição de código, entre outros.