Desafios: hora da prática

Domine o Controle de Versão com estes desafios práticos! De visualizar e modificar commits a sincronizar com repositórios remotos, estes **exercícios não obrigatórios** proporcionarão uma experiência prática para fortalecer sua compreensão do Git.

A prática constante é a chave para se tornar um mestre no Git!

Desafios

1. Acesse todos os commits realizados
2. Modifique o último commit
3. Reverta mudanças no repositório local
4. Apague um ou mais commits
5. Sincronize as modificações com o repositório remoto
6. Analise as mensagens de commits realizados e faça as alterações de acordo com as boas práticas

---

Caso precise de ajuda, opções de solução das atividades estarão disponíveis na seção “Opinião da pessoa instrutora”.

Opinião do instrutor

Desafio 1: Veja todos os commits realizados

Para visualizar todos os commits realizados em um repositório Git, você pode usar o comando git log. Este comando exibe o histórico de commits, mostrando informações como o hash do commit, autor, data, e mensagem do commit. Aqui estão algumas variações do comando git log que podem ser úteis:

- Para exibir o histórico completo: `git log`

- Exibir Alterações Detalhadas: `git log -p`
- Exibir Apenas Mensagens de Commit: `git log --oneline`
- Desafio 2: Modificar o Último Commit com `git commit --amend`:
- Faça um commit com algumas alterações no seu código.
- Perceba que esqueceu de incluir algumas modificações.
- Para adicionar as modificações esquecidas ao último commit, sem criar um novo commit, utilize o comando: `git commit --amend`

Desafio 3: Reverter Mudanças de um Commit com Git Revert:

- Crie um novo arquivo no seu repositório.
- Realize um commit adicionando esse novo arquivo.
- Para reverter automaticamente as mudanças feitas no último commit sem excluir o histórico: `git revert <hash-do-commit>`
- Se você precisar reverter uma série de commits, pode utilizar o seguinte comando: `git revert -n <hash-do-ultimo-commit-a-manter>`
- Lembre-se que após reverter, é necessário realizar um novo commit para aplicar a reversão ao repositório

Lembre-se de que o git revert é uma maneira segura e não destrutiva de desfazer alterações, pois não reescreve o histórico existente. Ele é particularmente útil em ambientes de trabalho compartilhados, onde reescrever o histórico pode causar problemas para outros colaboradores.

Desafio 4: Apagar um Commit com Git Reset:

- Faça uma série de commits com alterações no seu código.
- Escolha um commit específico que deseja excluir.
- Use para apagar o commit selecionado, desfazendo automaticamente as mudanças no código: `git reset --hard <hash-do-ultimo-commit-a-manter>`
- Se você apenas deseja desfazer commits, mas manter as alterações no diretório de trabalho, você pode usar `git reset --soft`.

Observação: No cotidiano de trabalho em desenvolvimento, a escolha entre `reset` e `revert` deve ser tomada com base nas necessidades específicas do seu projeto e na colaboração com as outras pessoas desenvolvedoras. Se você deseja desfazer alterações em um commit específico sem reescrever o histórico, `git revert` é uma escolha mais segura. Se você precisa reverter completamente para um estado anterior, `git reset` pode ser apropriado, mas use-o com cautela, especialmente em branches compartilhadas.

Desafio 5: Sincronize as modificações com o repositório remoto

- Execute o comando: `git push`

Desafio 6: Analise as mensagens de commits realizados e faça as alterações de acordo com as boas práticas:

- Para visualizar todo o histórico de commits, execute o comando: `git log`
- Leia a documentação da *Conventional Commits* para elaborar as mensagens de commits de acordo com as boas práticas
- Reescreva os commits utilizando os comandos já aprendidos: `git commit –amend` ou `git revert`.