Princípios de boas práticas de versionamento de código

Aula 2

Professor Gustavo Dias

Mesclar códigos usando Git

No desenvolvimento de software, várias pessoas costumam trabalhar ao mesmo tempo. O **Git**, uma ferramenta de controle de versão, tem recursos para **unificar mudanças** num código-base comum, sendo o **merge** e o **rebase** as **estratégias mais usadas.**

Mesclar códigos utilizando **merge** e **rebase**.

Definir a **importância dos branches** no desenvolvimento de trabalhos paralelos e isolados.

Praticar rebase e merge por meio de uma atividade.

***Rebase* para manter um histórico linear**

- O ***rebase*** é uma maneira de integrar as alterações, pegando todas as mudanças que foram aplicadas a um Branch e aplicando-as a outro.

- Equipes optam pelo ***rebase*** porque ele conserva um histórico de ***commits*** limpo e linear. Além disso, quando um ***Branch*** de funcionalidades é integrado ao ***Branch*** principal, todas as mudanças são aplicadas diretamente na ***main***, como se tivessem sido feitas diretamente nela.

Exposição

Cautelas ao rebasar

Embora o *rebase* possa ser uma ferramenta poderosa, ele tem suas armadilhas. Vamos conhecê-las!

© Getty Images

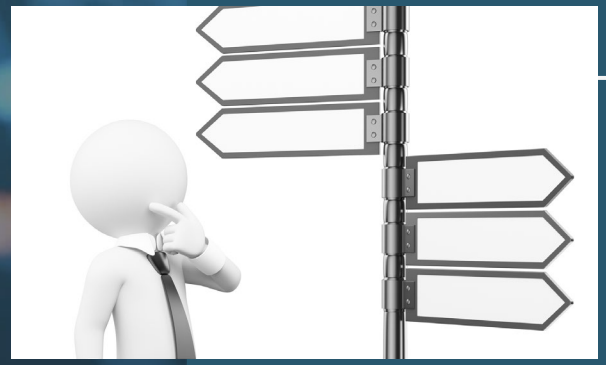
Alteração do histórico

Em *branches* compartilhados, fazer um *rebase* pode criar discrepâncias e confusões.

© Getty Images

Complexidade em conflitos

Ao surgirem conflitos durante o rebase, é necessário resolvê-los para cada commit individualmente, em vez de uma única vez, como no merge.



© Getty Images

Perda de contexto

O *rebase* cria um histórico linear, por isso perde-se o contexto de quando e em que *branch* um determinado trabalho foi realizado, dificultando a depuração e o rastreamento de alterações.

Quando usar *merge*?

O **merge** pega todas as alterações de um branch e as combina com outro branch em um novo **"commit de merge"**. Isso preserva o histórico de ambos os *branches* e claramente mostra quando e onde as alterações foram mescladas.

Ao contrário do rebase, **preserva o contexto e a temporalidade** das alterações, o que pode ser valioso para entender o histórico do projeto.

Dica:

Se a equipe valoriza um registro completo e detalhado de todas as alterações e de quando foram feitas, o *merge* pode ser a melhor opção.