



100%

ATIVIDADES  
6 DE 9FÓRUM DO  
CURSOVOLTAR  
PARA  
DASHBOARDMODO  
NOTURNOABRIR  
CADERNO

## Transcrição

Pode acontecer de não querermos que determinado arquivo seja monitorado, como no caso de um arquivo de configurações da IDE. Como poderemos fazer para que o Git o ignore?

Existe um **arquivo especial do Git**, chamado `.gitignore`, e todas as linhas que estiverem nele serão lidos e ignorados pelo Git. Se temos um arquivo denominado `ide-config` que queremos que seja ignorado, por exemplo, basta o incluirmos em `.gitignore`, digitando `ide-config` simplesmente. Da mesma forma, se tivéssemos uma pasta `ide`, incluiríamos `ide/`, em uma nova linha.

Porém, antes de conferirmos isto com `git status`, precisaremos adicioná-los, com `git add .gitignore`, por exemplo, e `git commit -m "Adicionando .gitignore"`. Neste momento, poderemos nos perguntar: em que momento criamos um commit? Apenas no fim do projeto? Quando finalizarmos tudo? Ou a cada linha modificada?

Este é um assunto muito extenso, que gera discussões bem calorosas, mas um consenso geral é que **jamais devemos commitar código que não funciona**. Isto é, o código deve estar sempre no estado funcional para ser commitado. Isto não significa que ele deva ser commitado apenas ao fim do projeto. A recomendação é que se gere um commit após cada alteração significativa.

Existem pessoas que defendem que o commit deve ser gerado ao fim do expediente, outras que dizem que isto deve ser realizado a cada alteração, **não existe uma regra**, e sim recomendações. Sempre que uma pequena funcionalidade for implementada, ou um bug for corrigido, é possível realizar um commit, para que no fim do dia, um conjunto de commits gere o sistema como um todo, e não um único commit.

Já entendemos o que é um repositório e como funciona seu conceito, inclusive transformamos nossa pasta em um repositório Git. Além disso, aprendemos a visualizar o seu status, como adicionar e salvar arquivos nele, visualizar as alterações feitas no projeto, e deixar de monitorar determinados arquivos ou pastas.

Conseguimos começar a trabalhar de forma bem interessante com o controle de versões. Mas como será que passamos a trabalhar em equipe, compartilhando o projeto usando um repositório Git?



304.0k xp

a