03

Vendo o histórico

Transcrição

Anteriormente, ficamos com a dúvida: como poderemos verificar o histórico de alterações, cada mensagem de commits feitos, o andamento do nosso projeto? O comando que poderemos utilizar para isto é `git log`, que nos mostrará diversas informações, sendo o primeiro deles um **hash do commit**, uma identificação única de cada commit, isto é, **não existem dois commits com o mesmo hash**.

Assim, conseguiremos realizar algumas manipulações, que veremos mais adiante. A informação seguinte se refere ao **branch**, ou "ramo" em que o commit se encontra. Neste caso, verificamos que há `HEAD` e `master`. Isto quer dizer que `HEAD` é o local onde nos encontramos, no nosso código, onde acontecem as alterações que fizermos, e que estamos em um ramo denominado `master`.

Além disso, temos a autoria do commit, e-mail configurado, data de commit, e mensagem. Mas como é que o Git sabe que este e-mail é o seu? Eu já tinha utilizado o Git algumas vezes neste computador, então algumas configurações já estavam definidas, o que é possível fazermos a partir do comando `git config --local` para cada projeto, ou, para a máquina toda, utilizando o `git config --global`.

Por enquanto, modificaremos as configurações para este único projeto, ou seja, as configurações definidas por meio deste comando só serão válidas para este repositório. Como anteriormente só foi exibido meu e-mail, configuraremos o nome, com `git config --local user.name "Vinicius Dias"`, após o qual pressionaremos "Enter".

VOLTAR AO TOPO

Poderemos visualizar as configurações salvas por meio de `git config user.name`, ou `git config user.email`. Então, os commits que fizermos a partir daqui terão este nome. Mas será que existe alguma alternativa ao `git log`?

Sim, existem várias! Uma das mais comuns nos permite visualizar todos os commits, sendo que cada uma ocupa uma única linha: `git log --oneline`. E se em vez de menos informações quisermos mais, como as alterações do commit, usaremos `git log -p`. O formato em que elas são exibidas conta com a versão anterior em vermelho, e a versão modificada logo abaixo, em verde.

Existe uma infinidade de formatos que podemos usar como filtros para mostrar nosso histórico, e em git log cheatsheet (http://devhints.io/git-log) há vários delas. Como exemplo, testaremos `git log --pretty="format:%H"`, comando que nos traz apenas o hash. O comando `git log --pretty="format:%h %s"`, por sua vez, traz o hash resumido seguido pela mensagem do commit. Assim, podemos gerar o histórico da nossa aplicação em formatos personalizados.

Aqui no curso estamos usando o VS Code — é possível utilizar qualquer editor de sua preferência —, mas imaginemos que estejamos usando uma IDE que cria uma pasta contendo configurações, os quais não queremos que o Git fique monitorando. De que forma podemos informá-lo disto? Veremos a seguir!

VOLTAR AO TOPO