**Git**

Repositório do Git seja inicializado, e para tal usamos o comando git init.

**Como fazemos para o Git passar a enxergar determinada pasta como um repositório e a observar as mudanças em seus arquivos?**

O git init inicializa um repositório no diretório em que o comando for executado. A partir deste comando, o Git poderá gerenciar as modificações realizadas nos arquivos.

Um comando que mostra o estado do nosso repositório, ou seja, quais arquivos foram alterados, ou não, é o git status

Nesta aula, aprendemos:

O que são (e para que servem) sistemas de controle de versões e como eles podem ajudar o nosso fluxo de desenvolvimento

Nos ajudam a manter um histórico de alterações;

Nos ajudam a ter controle sobre cada alteração no código;

Nos ajudam para que uma alteração de determinada pessoa não influencie na alteração realizada por outra;

Etc.

**Antes de qualquer interação com o git, você precisa informar quem é você para que ele armazene corretamente os dados do autor de cada uma das alterações no código.**

**No vídeo eu não fiz isso pois o git já estava configurado na máquina, mas para você fazer isso na sua, caso esteja começando a utilizar o git agora, basta digitar os seguintes comandos (estando na pasta do repositório git):**

**git config --local user.name "Seu nome aqui"**

**git config --local user.email** [**seu@email.aqui**](mailto:seu@email.aqui)

**Ao executar o comando git status, recebemos algumas informações que talvez não estejam tão claras, principalmente quando nos deparamos com termos como HEAD, working tree, index, etc.**

**Apenas para esclarecer um pouco, visto que entenderemos melhor o funcionamento do Git durante o treinamento, seguem algumas definições interessantes:**

**HEAD: Estado atual do nosso código, ou seja, onde o Git nos colocou**

**Working tree: Local onde os arquivos realmente estão sendo armazenados e editados**

**index: Local onde o Git armazena o que será commitado, ou seja, o local entre a working tree e o repositório Git em si.**

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.

**Como podemos definir o nome da pessoa que executa commits no repositório local atual?**

git config --local user.name "Nome da pessoa"

Alternativa correta! Assim todos os commits executados neste repositório serão atribuídos à pessoa com nome Nome da pessoa. Para mais detalhes e outras configurações possíveis (até algumas mais avançadas), você pode conferir este link: <https://git-scm.com/book/en/v2/Customizing-Git-Git-Configuration>**.**

**Conforme vimos no último vídeo, podemos visualizar o histórico de alterações em nosso projeto de forma muito fácil, através do comando git log.**

Apesar de ser fácil, este comando é muito poderoso. Execute git log --help e veja algumas das opções possíveis. Para alguns exemplos mais fáceis de entender, você pode pesquisar sobre git log ou dar uma olhada neste link: <https://devhints.io/git-log>.

**Devemos gerar um commit sempre que a nossa base de código está em um estado do qual gostaríamos de nos lembrar. Nunca devemos ter commits de códigos que não funcionam, mas também não é interessante deixar para commitar apenas no final de uma feature. A**