**Git e Github**

O git é um sistema que permite que várias pessoas façam alterações num código e envie para um armazenamento remoto, não permitindo que outras pessoas que também tenham feito alterações possam mandar essas novas modificações antes de baixar as que a outra pessoa fez. Isso facilita a comunicação entre os desenvolvedores já que não tem a necessidade de colocar o código num pendrive e levar até os outros para que eles possam ver as modificações feitas e iniciar a partir daí. Todos podem fazer alterações ao mesmo tempo e todos ficam sabendo na hora o que foi alterado.

Além do Git existem outras plataformas que fazem isso, como a CVS, SVN. Mercurial...

Para saber mais do git status:

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 os 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.

Além disso, os possíveis estados dos nossos arquivos são explicados com detalhes neste link: <<https://git-scm.com/book/pt-br/v2/Fundamentos-de-Git-Gravando-Altera%C3%A7%C3%B5es-em-Seu-Reposit%C3%B3rio>.

Para saber mais de quando commitar:

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*.

Essa pode ser uma discussão sem fim e cada empresa ou equipe pode seguir uma estratégia diferente. Estude sobre o assunto, entenda o que faz mais sentido para você e sua equipe e seja feliz! :-D

**Comandos no bash do git ou no terminal:**

- *git init*: inicia o repositório na pasta em que está.

- *git status*: exibe algumas informações da pasta que estamos, como os arquivos que não estão sendo rastreados por mudanças, diz quantos commits esse repositório já tem...

- *git add <file>*: adiciona o arquivo no commit para começar a ser rastreado.

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

- *git config --local user.email “*[*seu@email.aqui*](mailto:seu@email.aqui)*”*: Esses dois comandos servem para identificar quem que está fazendo as alterações nessa pasta no computador, deixando registrado sempre. Sempre fazer quando instala o git. Deve ser feito na pasta em que iniciamos o git.

Se alterarmos o --local por --global ele fará as alterações/definições de nome e e-mail de usuário para todos os projetos.

Em --local ele muda/define apenas para aquele repositório específico que estamos trabalhando.

Utilizando o comando sem o --local e os dados em “ “, ele apenas exibe qual o nome de usuário ou o e-mail que está configurado.

- *git rm –cached <file>*: remove o arquivo que adicionamos com git add e está aguardando o commit para ser rastreado.

- *git commit -m “mensagem”*: faz o commit das novas alterações que foram feitas no repositório. O -m é para podermos digitar a mensagem a seguir. Pode ser algo simples, não devemos fazer uma documentação toda nessa mensagem. O git monitora linha a linha de tudo o que estamos fazendo o commit por conta disso, ao executar o comando ele diz até quantas linhas o arquivo que estamos commitando tem.

Temos que fazer todo o processo de adicionar e commitar com a mensagem sempre que fizermos uma modificação. Na dúvida, utilize o git status para verificar como estão as coisas.

- *git log*: exibe o histórico de commits que fizemos nesse repositório. Ele mostra o head em primeiro lugar, sendo onde estamos atualmente na linha do tempo dos commits. Ele mostra o autor do commit, data e hora e a descrição dele.

--oneline: na frente ele exibirá cada commit de forma resumida em uma única linha.

-p: mostra todas as alterações detalhadamente, inclusive o que foi acrescentado e/ou tirado do código.

Existe uma infinidade de formas que podemos visualizar o histórico: <http://devhints.io/git-log>

- .gitignore: arquivo que criamos e tudo o que colocamos dentro são os arquivos que o git não irá monitorar, ou seja, criamos esse arquivo e escrevemos nele o nome das pastas/ e arquivos que desejamos que o git não veja as mudanças e afins que fizermos neles. A partir do momento que criamos esse arquivo só precisamos commitar ele e tudo o que estiver dentro dele será ignorado.