GIT

“Commits” - São versões do código.

“Untracked Files:” – Arquivos que não foram adicionados ao controle de versão.

“Branch” – versões ou divisões (diferentes) do código que estão sendo versionadas de formas separadas. *“****Branch****, em tradução literal, significa “ramo”. ... Quando criamos um repositório, ele automaticamente é iniciado com a****branch master****, que é equivalente ao tronco da sua árvore, ou seja, é a parte principal do seu projeto.”*

git add “Nome do arquivo [que está em vermelho]” (*Porém, é melhor usar o “git add .” para salvar todos que ainda não foram salvos de uma vez só).*

**gitgnore** – *Para o GIT não monitorar os arquivos*.

**git push** – *Subir arquivos para o github (Enviar uma versão do código para o servidor na nuvem). Push = empurrar. Empurrar atualização de versão para um repositório.*

**git pull** *– Para puxar alterações de repositórios para o repositório. Pull=puxar.*

**git clone EndereçoDoGitHUB** – *Para clonar um repositório.*

**git status** – *Verificar o estado do repositório (Pasta).*

**git add .**

**git commit -m “informação do que é”** – *Cria versões do código.*

**git config –global user.email “emailexemplo@gmail.com”** - *Configurar qual é o usuário(email do github).*

**git config –global user.name “Nome de quem fez”** – *Para quando você enviar as versões do código que você fez, ter a identificação de quem fez.*

**git remote add origin** [**https://github.com/JiscoTorres/Aula\_01\_Git\_DS34.git**](https://github.com/JiscoTorres/Aula_01_Git_DS34.git) *- Para enviar o código do computador local para a nuvem (Site github).*

**git push --set-upstream origin master** *– Define para qual branch ele enviará o código, nesse caso, enviará a atualização para a branch master ( É para a Branch master que, inicialmente, sempre enviamos para ela primeiro).*

**git reflog** *– Para ver o histórico das versões que já foram salvas do código.*

***git reset --hard id da versão*** *- Serve para navegar ou voltar para outras versões do código.*

*Ex:*

Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

*Ele deu um git reflog, e pegou o ID que aparece em amarelo o qual seria o ID da versão. Nesse caso, o Id foi o 53b2dfb. Com isso, ele voltará para essa versão.*

Enviar atualizações do código para o controle de versão

git status (Se não der certo, fazer o “git add.” Primeiro e depois colocar o “git status”)

git commit -m “….”

git push

Que seria igual a:

**git status**

**git add .**

**git status**

**git commit -m “...”**

**git push**

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamenteControle de versão

Git + Github

-No git, ocorre o controle de versão.

-No github, você poderá fazer a “arquivação” das versões. -> não é um instrumento de trabalho, e sim uma rede social.

Método para ter um controle de versão de determinado projeto, assim podendo trabalhar em equipe e discutir as versões do projeto.

Para fazer todo esse controle de versionamento, a “nuvem” que iremos guardar o código será o github.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

BRANCHES

*Explicação da professora*:

Para criar um branch: **git branch funcionalidade\_x**

Para acessar um branch: **git checkout funcionalidade\_x**

Para retornar para o branch master: **git checkout master**

Para remover um branch: **git branch -d funcionalidade\_x**

Para enviar o branch para o repositório: **git push origin <funcionalidade\_x>**

**git branch staging –** *Nome que se dá para uma branch que recebe atualizações que ainda não estão testadas.*

**git branch staging + Enter**