GIT

https://itau.udemy.com/course/git-completo-do-basico-ao-avancado/learn/lecture/13897466#overview

Sobre o controle de versão

* Histórico de alterações
  + É possível restaurar versão anterior sem prejuízo
* Pequenos ou grandes projetos
  + Pode ser utilizado por frentes diferentes de trabalho
* Ramos (branches)
  + Diferentes níveis à partir de um mesmo ponto do projeto
  + Mesclar funcionalidades após testadas
* Rastreabilidade
  + Identifica quando e quem realizou a mudança no código em determinado momento

Tipos de Versionadores

Centralizado

Nesse modelo o servidor contém todo o histórico e foi padrão por muitos anos. Ex.: Subversion (SVN) criado em 2000, GIT criado no final de 2005

Diagrama

Descrição gerada automaticamente

Se o servidor não tiver a proteção de backup, etc., e ocorrer um problema no servidor, o projeto é impactado.

Distribuído

Nesse modelo o servidor e as máquinas contêm todo o histórico do versionamento.

Diagrama

Descrição gerada automaticamente

Toda máquina possui uma cópia do repositório

Mudanças são realizadas localmente

Não depende de uma única máquina

Informações sobre o funcionamento

STATUS

Texto

Descrição gerada automaticamente

Untracked files: arquivos não rastreados pelo GIT (ainda), para adicionalos ao versionamento, executar “git add <nome do arquivo>”

Texto

Descrição gerada automaticamente

No exemplo abaixo existe uma versão já adicionada do arquivo “commands\_GIT.txt” para ser commitada, no entanto, ainda há mudanças nesse mesmo arquivo que não foram adicionadas.

Texto

Descrição gerada automaticamente

Tela preta com letras brancas

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente



Logotipo

Descrição gerada automaticamente



Texto

Descrição gerada automaticamente

Estagios de um arquivo

Diagrama

Descrição gerada automaticamente

Untracked: arquivos ainda não rastreados

Unmodified: arquivo commitado

Modified: arquivo que após o commit, sofreu alguma alteração

Staged: arquivo após modificado, adicionado na área para realizar o commit

Para comparar as alterações realizadas:

Texto

Descrição gerada automaticamente

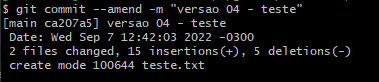
Para listar os commits

Os comits são listados em ordem decrescente

Texto

Descrição gerada automaticamente





Texto

Descrição gerada automaticamente

Movendo entre os commits

Texto

Descrição gerada automaticamente

DESFAZER Alterações no repositório

Diagrama

Descrição gerada automaticamente

GIT IGNORE

https://github.com/github/gitignore

Conceito de **FORK**

 O fork serve para realizar uma “copia” individualizando o projeto para dominio proprio sendo possível realizar alterações e contribuir (ou não) com o porjeto origem

Conceito de **PULL REQUEST**

 Proposta de mudanças que são encaminhadas como contribuição para o projeto origem qual foi realizado o fork/branch.

Conceito de **ISSUES / Labels / Milestone**

 Problema capturado por qualquer um, que é reportado e assim fica visível para comunidade toda que acompanha o projeto.

 Usado para categorizar os problemas capturados

 Usado para definir um determinado marco / backlog/ agrupamento de Issues

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

Conceito de **README**

Fornecer um Resumo/Documento/Apontamento incial do Projeto ou subdiretorios para que tenha maior facilidade de se obter informação sobre o repositorio.

<https://dillinger.io/>

Conceito de **SSH**

Utilizado para o GIT\_BASH conseguir realizar a comunicação com o REPO

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Nas configurações do REPO é possivel realizar o cadastro da chave SSH gerada.

A configuração é gerada por conta e não por repositório, uma vez gerada, não será necessário fazer para cada um dos repositorios existentes.

----------------

gerar chave SSH

----------------

ssh-keygen

cd ~/.ssh/

start ~/.ssh/

id\_rsa.pub <- chave gerada (caso pode ser renomeado e adicionado posteriormente)

eval $(ssh-agent)

ssh-add ~/.ssh/id\_rsa

Conceito de **BRANCHES**

Branch é uma ramificação no projeto que permite que funcionalidades sejam desenvolvidas separadamente sem impactar as funcionalidades estáveis no projeto.

Gráfico, Esquemático, Gráfico de bolhas

Descrição gerada automaticamente



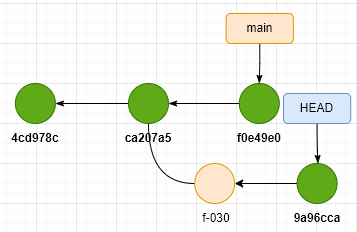


Diagrama

Descrição gerada automaticamente

Diagrama, Esquemático

Descrição gerada automaticamente



Texto, Tabela

Descrição gerada automaticamente

Conceito de **MERGE**

Integrar os fontes desenvolvidos em um ramo na branch atual. Ou seja, caso esteja na branch develop e execute o merge da branch main, significa que vai mesclar os dados da main para develop.

Mais natural o caminho de ter uma branch “PRODUCAO, HOMOLOGACAO E DESENVOLVIMENTO”; Na branch HOMOLOGACAO é realizado merge da DESENVOLVIMENTO e em PRODUCAO realizado o merge da HOMOLOGACAO.

O Merge trás as mudanças realizadas em determinado ramo para o ramo atual.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente com confiança média

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Ou seja, temos o ramo “develop” e o “main” no mesmo ponto independente do HEAD apontando para qualquer um que seja após o MERGE, diferente do ramo “testehead”

Conflitos

Os conflitos podem ocorrer quando existirem alterações em branchs que diferem em um mesmo arquivo.

Apendice A – Comandos

git config --global -l

git config --global user.name "Fulano de Tal"

git config --global user.email "fulano@de.tal"

git init