Algoritmos e Introdução à Programação (EAGS SIN 2020)

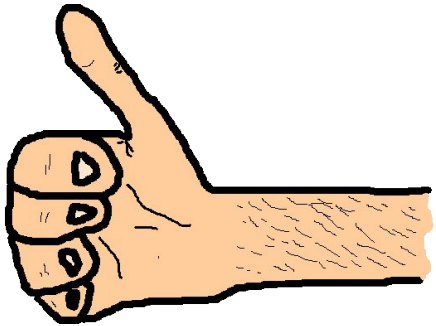
AULA 4.6 - Versionamento de código

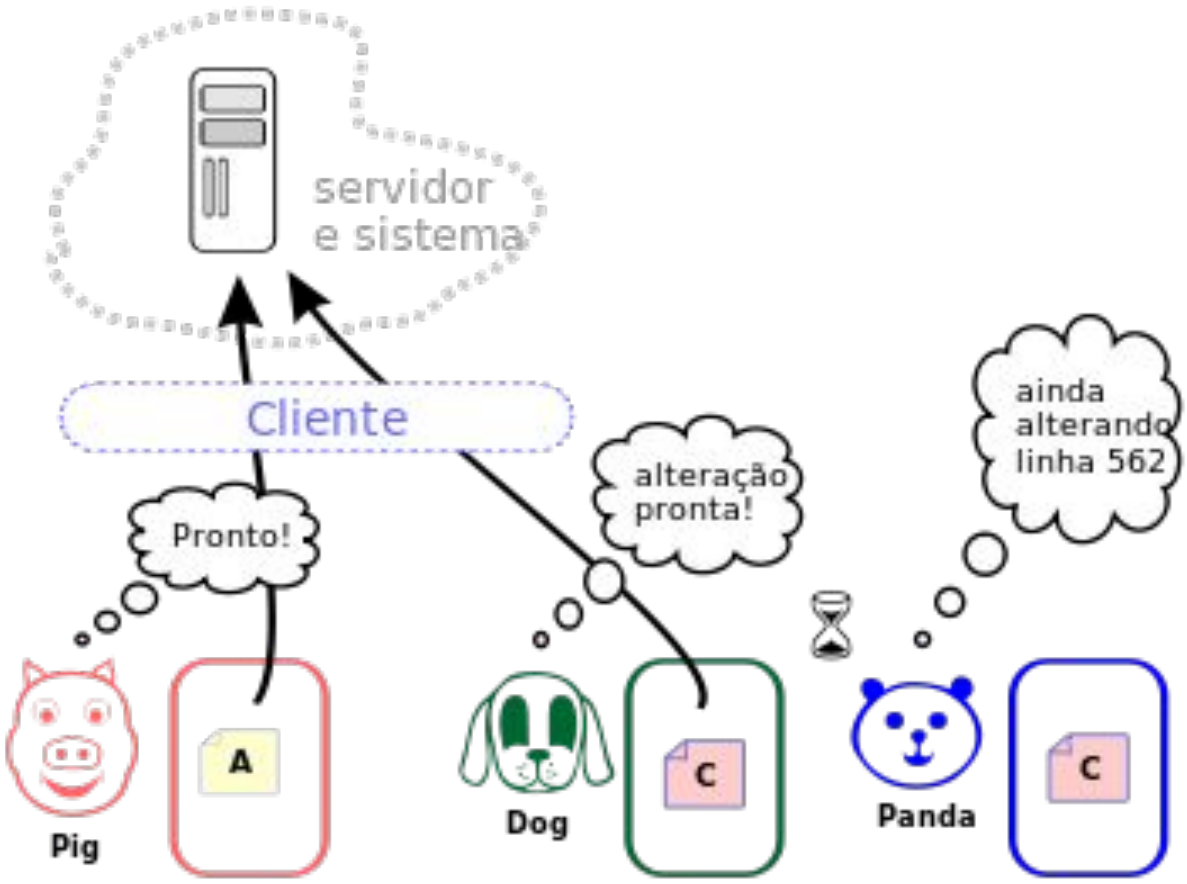
Apresentado por 2S SIN NETTO e 2S SIN MOURA

Se liga !

Você vai aprender:

1. O que é versionamento de código
2. Como funciona um ambiente de versionamento
3. Versionar projetos



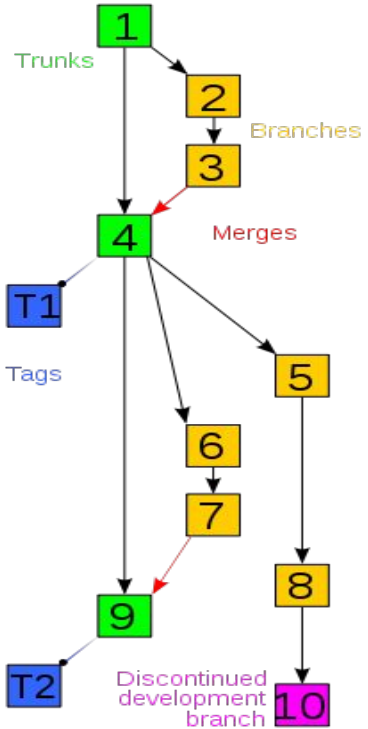

servidor
e sistema
Cliente
Pronto!
alteração
pronta!
ainda
alterando
linha 562
A
C
C
Pig
Dog
Panda

Cliente e servidor

A interação entre o código cliente (máquina local) e o código servidor (máquina servidor) se dá através de alguns passos importantes:

- Um **repositório** (no servidor para salvar os arquivos versionados) precisa existir
- O programador então cria um **clone** em seu ambiente de trabalho desse repositório para desenvolver seus arquivos localmente
- Toda vez que o programador concluir seu trabalho, ele conclui a edição dos arquivos do trabalho e gera uma **versão** para esse trabalho.
- Se as alterações realizadas nos arquivos do projeto forem suficiente e necessárias que todos do projeto tenham acesso a elas, o programador as **submete** para o servidor que se encarrega de **integrar** a versão recebida com a versão que já existia no servidor.

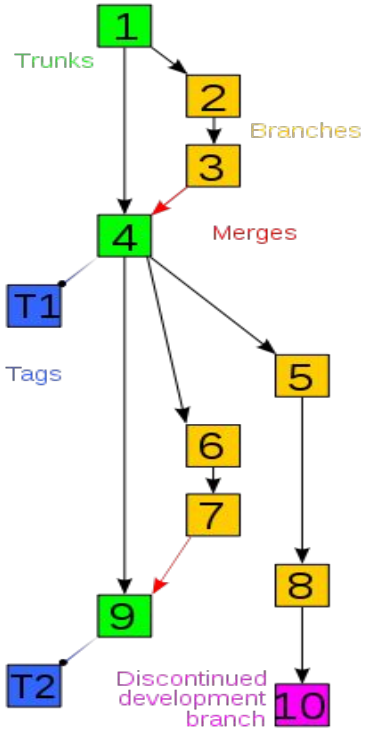
Ambiente de versionamento



A interação entre o código cliente (máquina local) e o código servidor (máquina servidor) se dá através de alguns passos importantes:

- **Trunk:**
  - "Tronco". É a estrutura única (cópia) existente na máquina local do desenvolvedor na qual se integram os diferentes "branches" da estrutura
- **Branche:**
  - É uma "ramificação" do tronco. Utilizado para modificar arquivos sem comprometer a versão dos arquivos do Trunk
- **Merge:**
  - É o processo de "mesclagem" de versões. Atualiza um branch com as alterações contidas no outro.

Ambiente de versionamento



➔ **Tags:**

- É a versão que “marca” uma integração entre dois branches. Essa tag é importante para encontrar facilmente as alterações feitas no projeto para o desenvolvimento de uma tarefa (feature) específica

➔ **Discontinued development branch**

- A criação de um branch além do trunk deve ser pontual e deve gerar uma previsível “mesclagem” - *merge* - entre os branches. Uma vez que o branche cumpriu seu papel, outro deverá ser criado

Histórico de envio

- **Commit**
  - Toda alteração realizada na máquina local do desenvolvedor e dada como funcional deve gerar um histórico de alterações que será identificado por *commits.*
- **Push**
  - Ao término de um ou mais *commits,* ou seja, quando o desenvolvedor terminar de alterar tudo o que é necessário alterar ele precisa enviar para o servidor as alterações para que os demais programadores tenham acesso a alas.
- **Update**
  - Um programador realiza uma atualização do seu ambiente de trabalho para receber as alterações do projeto realizadas pelos colegas

Estrutura de branches

- **Develop**
  - Brunch principal de desenvolvimento do projeto na máquina local do programador.
- **Master**
  - É o brunch que possui a versão de projeto mais atualizada. É essa versão que é copiada para a máquina do desenvolvedor no momento em que um *clone* é realizado criando uma cópia de trabalho local. A versão desse branch deve ser exatamente a mesma que está em funcionamento no ambiente de produção.
- **Outros**
  - Outros branches podem ser criados durante o ciclo de vida do projeto, mas ao término de cada branche deve haver uma integração com o branch Develop antes de serem integrados com o branch Master.

AULA PRÁTICA

Obrigado !

Você aprendeu:

1. O que é versionamento de código
2. Como funciona um ambiente de versionamento
3. Versionar projetos