



Usabilidade, desenvolvimento web, mobile e jogos

Felipe Andrade

(@prof.unibh.br)

Rafaela Moreira

(rafaela.cruz@prof.unibh.br)



PARA PENSAR...

- ★ Você está trabalhando em um projeto com mais 3 desenvolvedores ao mesmo tempo e precisa repartir as novas funcionalidades do sistema com eles.
- ★ Como você faz a junção do código ao final do desenvolvimento? (Lembrando que o mesmo arquivo pode ser editado por mais de um dev)



PARA PENSAR...

- ★ Seu chefe pediu para você deletar uma funcionalidade do sistema que não é utilizada. Após 3 meses ele decidiu que quer essa funcionalidade de volta.
- ★ Como você faz para recuperar essa funcionalidade e implementá-la no projeto mesmo após muitas outras mudanças?



PARA PENSAR...

- ★ Você é muito otimista e por isso não possui nenhum sistema de backup automático da sua aplicação em ambiente de desenvolvimento. Um belo dia seu computador queima e você não havia copiado suas últimas features para um pendrive.
- ★ Como você faz para recuperar esse código?



SISTEMA DE CONTROLE DE VERSÃO

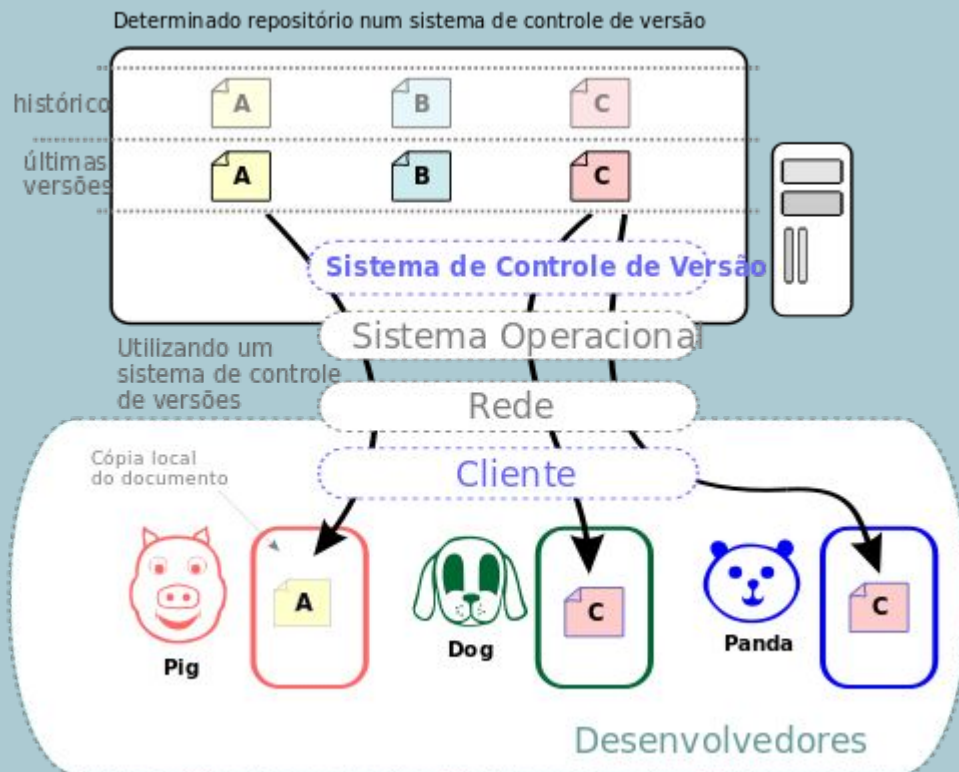
- ★ Software que tem a finalidade de gerenciar diferentes versões no desenvolvimento de um documento qualquer.
- ★ Esses sistemas são comumente utilizados no desenvolvimento de software para controlar as diferentes versões — histórico e desenvolvimento — dos códigos-fontes e também da documentação.



CONTROLE DE VERSÃO - VANTAGENS

- ★ Controle do histórico
- ★ Trabalho em equipe
- ★ Marcação e resgate de versões estáveis
- ★ Ramificação de projeto
- ★ Segurança
- ★ Rastreabilidade
- ★ Organização
- ★ Confiança

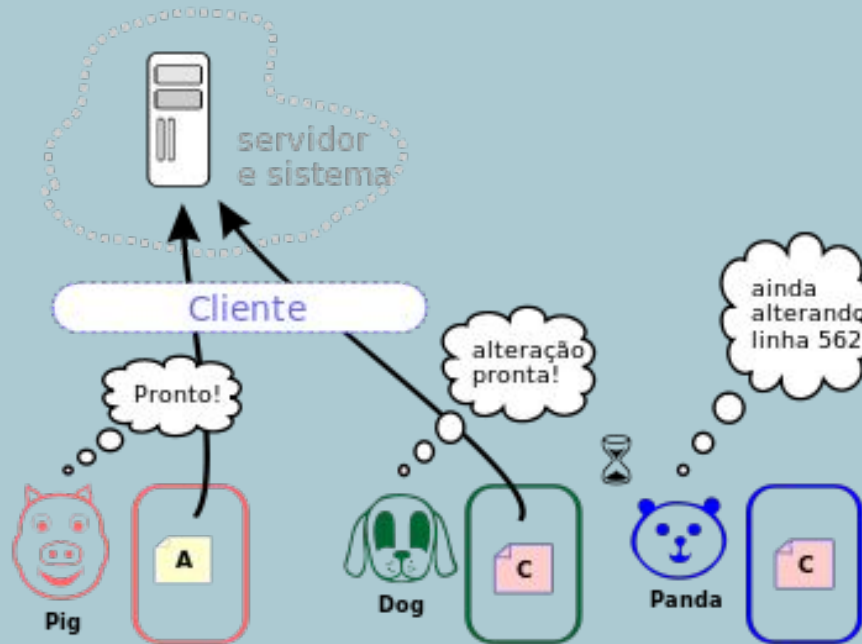
CONTROLE DE VERSÃO - FUNCIONAMENTO



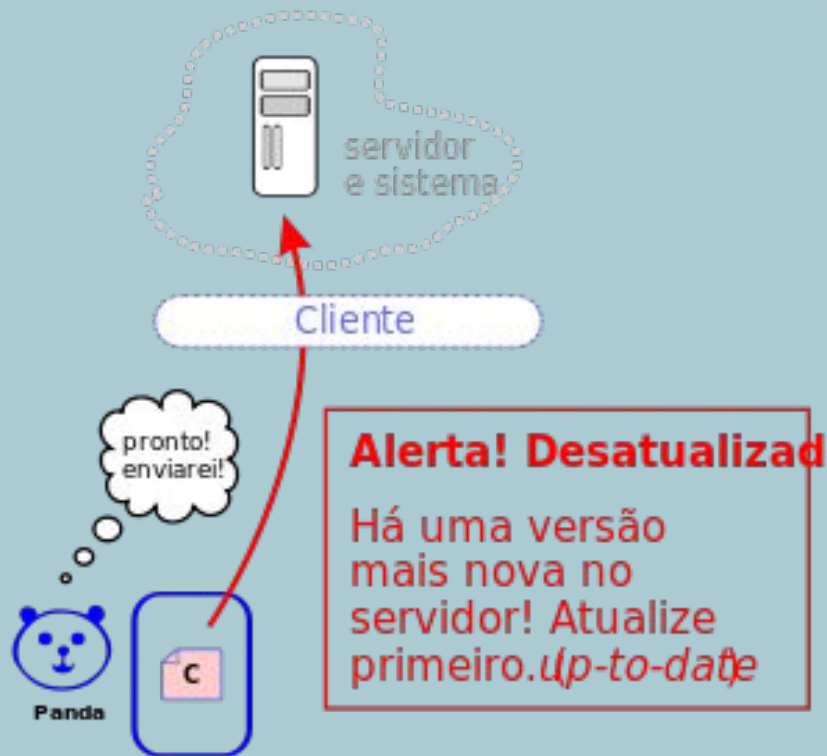
CONTROLE DE VERSÃO - FUNCIONAMENTO



CONTROLE DE VERSÃO - FUNCIONAMENTO



CONTROLE DE VERSÃO - FUNCIONAMENTO



CONTROLE DE VERSÃO - FUNCIONAMENTO



CONTROLE DE VERSÃO - FUNCIONAMENTO



VERSIONAMENTO DE SOFTWARE COM GIT



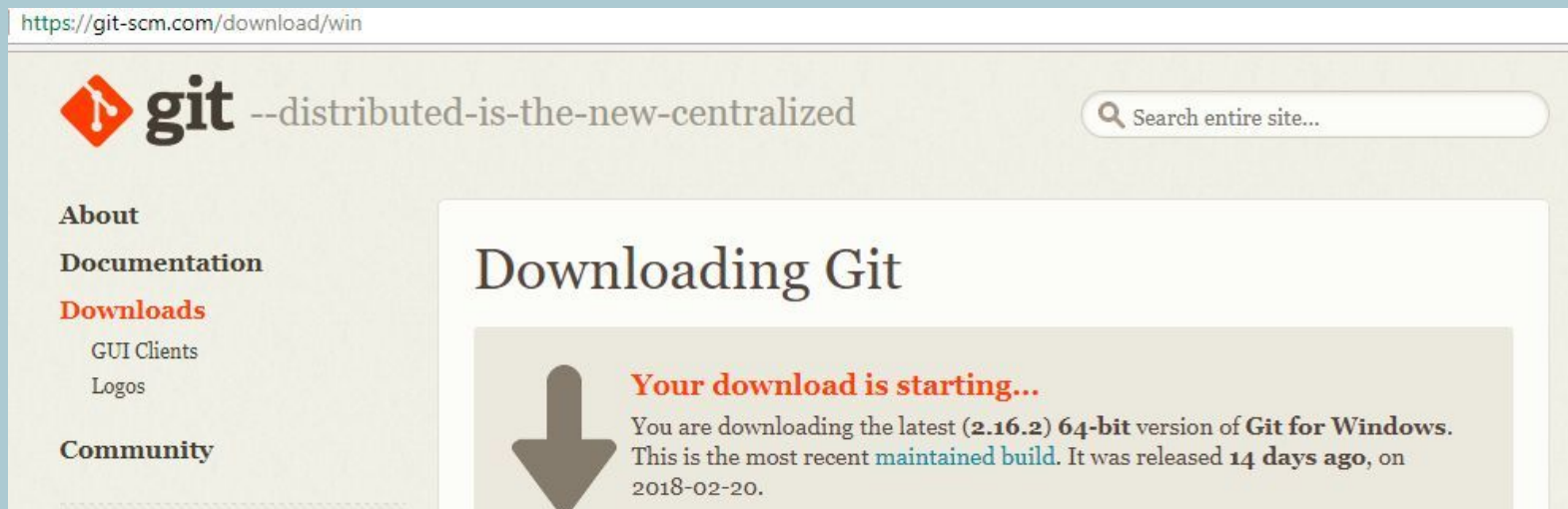
★ Git:

- Sistema de controle de versão distribuído
- Sistema de gerenciamento de código fonte.
- Desenvolvido por Linus Torvalds (criador do Linux)

★ Utilizado para criação de um histórico de alterações em código fonte.

Baixar o Git

★ Link do site: <https://git-scm.com/download>



CONCEITOS IMPORTANTES

★ REPOSITÓRIO

- Local onde os arquivos e suas cópias serão armazenados.
- Local/Remoto

★ BRANCH

- Ramos, cópias do código original .
- Permite alterações de forma segura, sem afetar as funcionalidades.

★ MERGE

- Fundir a cópia com o ramo principal.

CONCEITOS IMPORTANTES

★ PUSH REQUEST

- Envio de modificações para o repositório central.

★ PULL REQUEST

- Obtenção das modificações para a sua máquina.

★ FORK

- Cópia de um repositório remoto para a máquina local.
- Pegar um código público e utilizá-lo.

COMANDOS BÁSICOS

★ Configuração de Usuário (assinatura)

- Permite manter a rastreabilidade dos arquivos ao criar/alterar um arquivo.
- Abrir o cmd ou git-bash e digite:

```
$ git config --global user.name "Nome"  
$ git config --global user.email "nome@email.com"
```

- Conferir se deu tudo certo:

```
$ git config --global list
```

COMANDOS BÁSICOS

★ Criando repositório local

- Crie uma pasta e adicione um arquivo “teste.txt”
- Navegue até o diretório através do cmd

```
$ git init
```

- Saída esperada:

```
Initialized empty Git repository in  
caminho/.git
```

COMANDOS BÁSICOS

★ Verificar se está versionado

```
$ git status
```

- Saída para repositório vazio:

```
On ramo master
```

```
No commits yet
```

```
Arquivos não monitorados:
```

```
(utilize "git add <arquivo>..." para  
incluir o que será submetido)
```

```
teste.txt
```

COMANDOS BÁSICOS

- ★ Adicionar o arquivo “teste.txt” ao repositório

```
$ git add teste.txt  
$ git add --all
```

- ★ Confirmar inclusão

```
$ git commit teste.txt
```

COMANDOS BÁSICOS

★ Visualizar histórico de alterações

```
$ git log  
$ git log --stat
```

★ Mudanças entre commit

```
$ git diff numerocommit1 numerocommit2
```

COMANDOS BÁSICOS

- ★ Recuperar versão específica do arquivo

```
$ git checkout numerocommit
```

- ★ Voltar para última versão (HEAD)

```
$ git checkout master
```

COMANDOS BÁSICOS

★ Desfazendo alterações

```
$ git checkout teste.txt
```

★ Desfazer todas as alterações

```
$ git reset --hard
```

IGNORAR ARQUIVOS

- ★ Criar na pasta do arquivo, um arquivo com o nome .gitignore e adicione as extensões que não deseja rastrear.

```
*.exe
```


CLONAR REPOSITÓRIOS

★ Criar uma nova pasta fora do repositório

- Acesse a pasta no prompt e digitar

```
$ git clone diretorioRepositorio
```

BRANCH DEVELOP

- ★ **Master:** versão de produção.
- ★ **Developer:** versão em desenvolvimento + novas features



BRANCH DEVELOP

★ Criar branch

```
$ git branch develop
```

★ Visualizar branch

```
$ git branch
```

★ Selecionar branch

```
$ git checkout develop
```

UNIÃO DE CODELINES

★ Levar as alterações para a master

```
$ git checkout master  
$ git merge develop
```

★ Enviar para o servidor

```
$ git push
```

GITHUB

- ★ Serviço web que oferece diversas funcionalidades extras aplicadas ao **git**.
- ★ Hospedar projetos.
- ★ Mundialmente usado.
- ★ Mais de **56 milhões** de usuários ativos mundialmente contribuindo em projetos comerciais ou pessoais.
- ★ Abriga mais de **100 milhões de projetos** alguns deles que são conhecidos mundialmente.



GitHub

unibh>


GITHUB

- ★ Criar uma conta no GitHub
- ★ Criar um novo repositório no GitHub

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner * Repository name *

 rafapcmor ▼ /

Great repository names are short and memorable. Need inspiration? How about [expert-happiness?](#)

- ★ Copiar a URL do repositório

GITHUB

★ Copiar a URL do repositório

★ Clonar repositório

```
$ git clone urlRepositorio
```

★ Adicionar repositório

```
$ git remote add origins urlRepositorio
```

DÚVIDAS?



TUTORIAL

★ https://rogerdudler.github.io/git-guide/index.pt_BR.html

