Aula 3

Fundamentos de Design de Sistemas

Prof. Vinicius Pozzobon Borin

Conversa Inicial

- **O objetivo de hoje é conhecer o Git e o controle de versionamento de arquivos**

- **Veremos:**
  - **O que é Git e versionamento**
  - **Repositórios Git**
  - **Termos do Git**
  - **Comandos Git**

Git

Motivação

Fonte: Borin, 2021

Motivação

- O objetivo do Git é resolver esse tipo de problema
- Ele realiza o controle de versionamento, seja de documentos, seja de softwares

Motivação

- O Git também auxilia no trabalho colaborativo, pois mantém a informação de qual pessoa editou

Motivação

- Graças ao Git, é possível restaurarmos um software para uma versão mais antiga caso ele tenha sido publicado com problemas

Ferramenta Git

Postmodern Studio / Shutterstock

Repositório

- É o local em que ficam os arquivos do projeto, que serão controlados por versionamento

Git SCM

- Download
  - <https://git-scm.com/>
- Windows somente
  - <https://gitforwindows.org/>
- Guia prático
  - <https://rogerdudler.github.io/git-guide/index.pt_BR.html>

Instalação

- Trataremos da instalação do Git em aulas futuras, tanto em ambiente Windows quanto em Linux

Configurando o Git

git config

- Define o usuário que irá trabalhar com o Git
- Sintaxe:
  - git config --global user.name "Vinicius Borin"

git config

- Define o e-mail que irá trabalhar com o Git
- Sintaxe:
  - git config --global user.email "vinicius@email.com"

git config

- Define o editor que irá trabalhar com o Git
- Sintaxe:
  - git config --global core.editor notepad

git config

- Listando as configurações
- Sintaxe:
  - git config --list

Iniciando o repositório Git

- Pelo *prompt* de comando, entre na pasta do seu repositório
- Por exemplo:
  - C:\Users\Vinicius\PycharmProjects\pythonProject

git init

- Sintaxe:
  - git init

Fonte: Borin, 2021

git init

Fonte: Borin, 2021

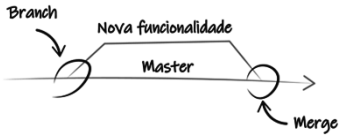
Terminologia Git

Branchs (ramificações)

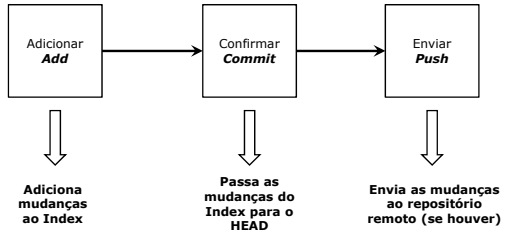
- Podemos ramificar nosso projeto para trabalharmos em funcionalidades separadas simultaneamente
- O *branch* padrão é chamado de *master*. É criado quando você cria o repositório

Branchs (ramificações)

- Podemos mesclar *branchs* para juntar partes desenvolvidas (*merge*)



Fluxo de trabalho



git add

- **Adiciona mudanças ao arquivo temporário Index**
- **Sintaxe:**
  - **git add <arquivo>**
  - **git add ***

git commit

- **Confirma as mudanças, gravando permanente no arquivo HEAD do repositório**
- **Sintaxe:**
  - **git commit –m "Comente suas modificações"**

git push

- **Envia as alterações do HEAD do repositório local para um repositório remoto**
- **Sintaxe:**
  - **git push origin master**
  - **git push origin funcionalidade_x**

Outros comandos Git

git log

- **Apresenta o histórico de *commits***
- **Sintaxe:**
  - **git log**

git status

- Mostra os arquivos alterados e também os novos ainda não monitorados
- Sintaxe:
  - git status

git branch

- Apresenta todos os *branchs* do projeto, bem como o que você está usando
- Sintaxe:
  - git branch

git reset

- Retorna a um estado anterior de *commit*
- Sintaxe:
  - git reset --tipo <ID_Commit>

git reset

- Tipo de *reset*:
  - *Soft* – retorna imediatamente ao estado antes do *commit*
  - *Mixed* – retorna ao estado antes do *commit* e antes do *add*
  - *Hard* – apaga tudo para antes do último *commit* (arquivos, alterações, tudo é apagado)

git diff

- Mostra os detalhes das alterações nos arquivos
- Sintaxe:
  - git diff
  - git diff --name-Only
  - git diff <nomeArquivo>

Trabalhando com *branchs*

git branch

- Cria um novo *branch*
- Sintaxe:
  - git branch <nomeBranch>

git checkout

- Altera o *branch* atual de trabalho
- Sintaxe:
  - git checkout <nomeBranch>

git checkout

- Podemos agora modificar o que quisermos dentro de outro *branch,* e o *master* permanecerá intacto