Versão 1

**============================================================**

**Índice**

============================================================

# [Gerenciar usuário](#Gerenciar_usuario)

# [Criar Repositório Git](#Criar_Repositorio_Git)

# [Gerenciar os Commits](#Gerenciar_os_Commits)

# [Ver detalhes dos Commits](#Ver_detalhes_dos_Commits)

# [Mover entre os Commits](#Mover_entre_os_Commits)

# [Desfazer Mudanças nos Arquivos](#Desfazer_Mudancas_nos_Arquivos)

# [Ignorando Mudanças nos Arquivos pelo Git](#Ignorando_Mudancas_Arquivos_Git)

# [Clonando Repositórios Locais](#Clonando_Repositorios_Locais)

**USANDO O GITHUB**

# [Clonando Repositórios do Github](#Clonando_Repositorios_Github)

# [Push, Enviando para o Repositório Github](#Push_Enviando_para_Repositorio_Github)

# [Pull, Baixar a última versão do repositório do Github](#Pull_Baixar_ultima_versao_repositorio)

# [Branch, Criar e Gerenciar um Branchs(galhos) do repositório](#Branch_Criar_Gerenciar_um_Branch)

# [Merge, Agrupar versões em uma](#Merge_Agrupar_versoes_em_uma)

**============================================================**

**============================================================**

## Gerenciar usuário

============================================================

**// Criar um usuário**

$ git config --global user.name "rafaelFreitas"

**// configurar o email do usuário**

$ git config --global user.email "rafaelfreitas.lima29@gmail.com"

**// para confirmar se foi criado**

$ git config --global user.name

*rafaelFreitas*

**// Lista todas as caracteristicas do usuário**

$ git config --global --list

*user.name=rafaelFreitas*

*user.email=rafaelfreitas.lima29@gmail.com*

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Criar Repositório Git ============================================================

**// Mostra os arquivos que estão na pasta**

$ dir -a

*. .buildpath .project composer.json index.php views*

*.. .htaccess .settings composer.lock vendor*

**// Para Criar um repositório Git**

rafael@pc-rafael MINGW64 /c/xampp/htdocs

$ git init

*Initialized empty Git repository in C:/xampp/htdocs/.git/*

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Gerenciar os Commits ============================================================

**// Para ver o status dos arquivos, se foram modificados ou se estão sendo rastreados**

$ git status

**// Adiciona o rastreio a todos os arquivos do diretório**

$ git add --all

**// Adiciona o rastreio a um arquivo**

$ git add nomeArquivo.extensão

**// Para commitar as alterações**

$ git commit -m "comentário da versão"

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Ver detalhes dos Commits ============================================================

**// Para ver o log de alterações**

$ git log

**// Para ver o que foi mudado entre commits**

// tem um espaço entre os códigos

$ git diff código\_do\_commit código\_do\_commit\_anterior

**// Para ver com mais detralhes o que foi modificado entre os commits**

$ git log --stat

// Para sai do git log é só apertar no "q"

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Mover entre os Commits ============================================================

MOVER-SE ENTRE OS COMMITS

**// mudar para um commit**

// $ git checkout 2b42496e00be55c37ce895c67a9e2f1324d73bc8

$ git checkout código\_do\_commit

// mudar para o commit master

$ git checkout master

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Desfazer Mudanças nos Arquivos ============================================================

**// Desfazer mudança em um arquivo (não apaga arquivos não rastreados pelo git)**

$ git checkout arquivo.extensão

// **Desfazer mudanças em vários arquivos** **(não apaga arquivos não rastreados pelo git)**

$ git reset --hard

// **Desfazer mudanças em vários arquivos (apaga arquivos não rastreados pelo git)**

// Mostra a lista de arquivos não rastreados e que vão ser apagados

$ git clean -n

// Apaga os arquivos listados

$ git clean -f

// **Desfazer mudanças em vários arquivos selecionados (apaga arquivos não rastreados pelo git)**

// Mostra a lista de arquivos não rastreados e que vão ser apagados

$ git clean -i

// Escolha o "s"

s

// Escolha o número dos arquivos que serão apagados pelo número com e aperte o "enter", após escolher aperte "enter".

// Para executar aperte "c" e após o "enter"

c

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Ignorando Mudanças nos Arquivos pelo Git ============================================================

**// Deve-se criar um arquivo dentro da pasta do projeto com o nome em banco com a extensão ".gitignore" para o linux ou ".gitignore." para o windows, pois ele não aceita nomes de arquivos em branco.**

// **Para ignorar um arquivo ou vário só e necessário colocar o nome do arquivo em um linha dentro do arquivo ".gitignore".**

// **Para ignorar um tipo de arquivo só é necessário colocar o "\*.extesão" , como exemplo " \*.html ", que ignorará todos os arquivos html que não foram rastreados pelo git.**

**// Tirar todos os arquivos do rastreamento do Git,** Pois não se pode ignorar arquivos já rastreados.

$ git rm -r --cached .

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Clonando Repositórios Locais ============================================================

**// Para clonar repositórios locais só é necessário utilizar o comando git clone caminho\_do\_repositorio\_a\_ser\_clonado, você deve estar na pasta onde o clone ficará.**

$ git clone C:/pasta/pasta/repositorio

ou

$ git clone C:/repositorio

// **OBS.: os arquivos ignorados pelo gitignore não serão clonados.**

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Clonando Repositórios do Github

============================================================

**// Para clonar repositório do Github só é necessário utilizar o comando git clone caminho\_do\_repositorio\_do\_github, você deve estar na pasta onde o clone ficará.**

$ git clone https://github.com/rafaelfreitaslima29/referencias.git

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Push, Enviando para o Repositório Github

============================================================

**// Para enviar para o repositório do Github é necessário utilizar o comando git push caminho\_do\_repositorio\_do\_github, e colocar seu login e senha do Github.**

$ git push https://github.com/rafaelfreitaslima29/referencias.git

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Pull, Baixar a última versão do repositório do Github

============================================================

**// Para receber a última versão dos arquivos do repositório do Github é necessário utilizar o comando git pull, que ele será atualizado.**

$ $ git pull

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Branch, Criar e Gerenciar um Branchs(galhos) do repositório

============================================================

**// Para criar uma versão alternativa do seu repositório Local ou Não, para trabalho em paralelo, é necessário utilizar o comando git branch nome\_da\_branch, que ela será criada.**

// Obs.: Ele cria mas não fica ativa.

$ git branch desenvolvimento

// cria e deixa ativa

$ git checkout -b desenvolvimento

**// Para ver todas as branchs do seu repositório Local. Utilize o comando git branch.**

$ git branch

**// Para ver todas as branchs do seu repositório no Servidor. Utilize o comando git branch -r .**

$ git branch -r

**// Para ver todas as branchs do seu repositório do Servidor e Locais. Utilize o comando git branch -a .**

$ git branch -a

**// Para mudar de branch, utilize o comando git checkout nome\_da\_branch.**

$ git checkout desenvolvimento

**// Para criar a branch no Github, após criar a branch no repositório local utilize o comando git push --set-upstream origin nome\_da\_branch ou git push -u origin nome\_da\_branch .**

$ git push --set-upstream origin desenvolvimento

ou

$ git push -u origin desenvolvimento

**// Para alterar o nome da branch. Utilize o comando git branch -m novo\_nome\_da\_branch para mudar o nome da branch atualmente ativa ou git branch -m nome\_da\_branch novo\_nome\_da\_branch para uma branch que não estiver ativa.**

$ git branch -m dev

ou

$ git branch -m desenvolvimento dev

**// Para não é possível alterar o nome da Branch do Github, é necessásio apagar a Branch que se quer alterar o nome e após criar uma nova com o nome novo. Para apagar a branch no Github. Utilize o comando git push --delete nome\_da\_branch.**

$ git push --delete dev

**// Para deletar um branch Local. Utilize o comando git branch -d nome\_da\_branch.**

$ git branch -d desenvolvimento

**// Para deletar um branch do Servidor Remoto. Utilize o comando git push origin --delete nome\_da\_branch.**

$ git push origin --delete desenvolvimento

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Merge, Agrupar versões em uma

============================================================

**// Para agrupar duas versões em uma é necessário estar no branch que você quer que receba as alterações. Para isso utilize o comando git merge nome\_da\_branch\_que\_você\_que\_as\_alteraçõesl, então a branch que você estiver será agrupada.**

// Estando na master

$ $ git merge desenvolvimento

// se não tiver conflitos ele fará um commit.

# [Índice](#indice)

**============================================================**

###############################################################

**============================================================**

Questões de Concursos

============================================================

**1 - Qual comando deve ser executado, em um repositório Git, através do terminal, para criar um branch chamado ufrpe e colocá-lo como ativo?**

A) git checkout ufrpe

B) git branch -d ufrpe

C) git checkout -b ufrpe

D) git pull ufrpe

E) git push ufrpe

Gabaritos: 1-c;

# [Índice](#indice)

**============================================================**

###############################################################