Guia do GitHub

1 O que é o Git ou GitHub

O **GitHub** é uma plataforma que permite o desenvolvimento de código por uma equipe de forma colaborativa, assim como garante o controle de versão do projeto. Ou seja, é possível acessar versões anteriores, verificar o que foi modificado, quem modificou e ainda o por que da modificação. O sistema não se restringe somente a códigos, mas pode ser usado para armazenar qualquer tipo de documento de projeto, como este guia.

2 Git no Windows

Diferente do Linux, que já vem com o **Git Bash** instalado, no Windows é preciso fazer sua instalação. O download é feito através do link https://git-scm.com/download/win. Após a conclusão do Download, deve haver um programa chamado Git Bash no sistema.

3 Git Bash

O **Git Bash** é a interface que usaremos do Git. Ele nada mais é que um terminal do Git no Windows e aceita todos os comandos do cmd. Existe uma interface gráfica para o Git, mas seu uso não é recomendado por não possuir muitas funcionalidades.
Também é possível acessar o Git Bash em uma pasta clicando com o botão direito em qualquer lugar dela e escolhendo a opção **Git Bash Here**.
Quando o Bash inicializar, faça a configuração inicial com os seguintes comandos, inserindo seu email e usuário, que podem ser cadastrados no site https://github.com/ .

```
Tiago Bachiega@Tiago-PC MINGW64 ~/Desktop
$ git config --global user.email tiago_almeida0297@hotmail.com

Tiago Bachiega@Tiago-PC MINGW64 ~/Desktop
$ git config --global user.name "TiagoB936"
```

4 Como baixar o repositório

Baixar, ou **clonar**, o repositório remoto para o seu computador pode ser feito usando-se o seguinte comando:

```
Tiago Bachiega@Tiago-PC MINGW64 ~/Desktop
$ git clone https://github.com/TiagoB936/GitTest.git
Cloning into 'GitTest'...
remote: Counting objects: 21, done.
remote: Total 21 (delta 0), reused 0 (delta 0), pack-reused 21
Unpacking objects: 100% (21/21), done.
```

Note que ele baixou o projeto **GitTest** para o seu computador. Para baixar qualquer repositório, basta substituir o link utilizado na chamada a **git clone** pelo link do repositório respectivo.
Entre na pasta baixada pelo comando.

```
Tiago Bachiega@Tiago-PC MINGW64 ~/Desktop
$ cd GitTest/
```

5 Organização do Git

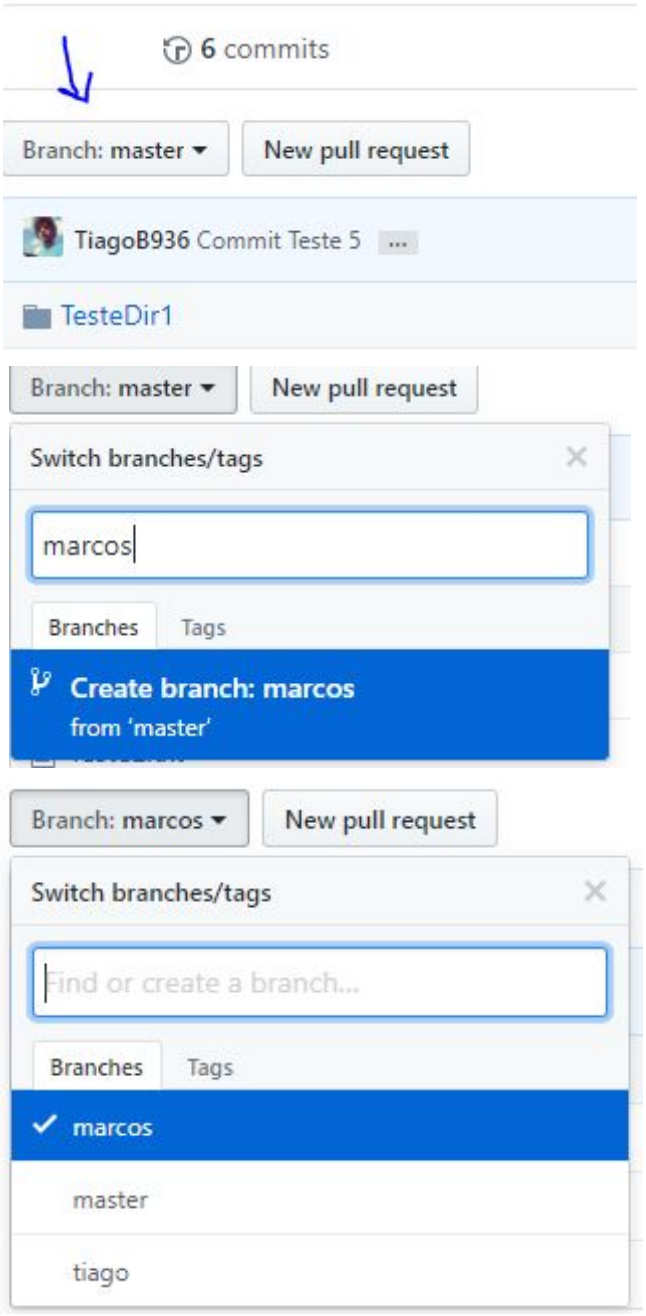
5.1 Branches

Uma **Branch** é uma linha de desenvolvimento do código. A branch principal do projeto é a **Master** e nela está a versão mais atual dos arquivos.

A branch pode ser **remota** ou **local**, correspondendo, respectivamente, à linha de desenvolvimento nos servidores do Git e no seu computador.

Cada desenvolvedor deve ter a sua branch e ela deve ter o nome de quem é seu dono, para facilitar o gerenciamento do projeto. Caso seja necessário criar outra branch para o mesmo usuário, deve-se criar com o nome **<usuário>-<número da branch>**. Por exemplo, a minha primeira branch deve se chamar **tiago** e, se necessário criar uma segunda branch, seu nome deve ser **tiago-2**.

Para criar uma **branch remota**, acesse o site do GitHub do projeto e siga os seguintes passos:



Por sua vez, para criar uma **branch local**, abra o Git Bash na pasta do projeto e digite o comando:

```
Tiago Bachiega@Tiago-PC MINGW64 ~/Desktop/GitTest (tiago)
$ git checkout -b marcos
Switched to a new branch 'marcos'
```

Para confirmar em qual branch você está trabalhando, use

```
Tiago Bachiega@Tiago-PC MINGW64 ~/Desktop/GitTest (marcos)
$ git branch
* marcos
  master
  tiago
```

5.2 Atualizando o repositório local

Sempre que iniciarmos o trabalho no projeto, devemos ter certeza de que estamos trabalhando na versão **mais atual**. Para isso, devemos **pegar** a versão mais atual da branch master do servidor remoto com o comando

```
Tiago Bachiega@Tiago-PC MINGW64 ~/Desktop/GitTest (marcos)
$ git pull origin master
From https://github.com/TiagoB936/GitTest
 * branch            master     -> FETCH_HEAD
Already up to date.
```

No caso, como eu já tinha a versão atual, foi mostrada a mensagem **Already up to date**.

5.3 Após fazer as alterações e enviando para o repositório

Após alterar os arquivos, devemos **adicioná-los** à lista de arquivos a serem enviados ao repositório remoto. Para incluir todos os arquivos modificados, devemos usar o comando

```
Tiago Bachiega@Tiago-PC MINGW64 ~/Desktop/GitTest (marcos)
$ git add *
```

Após isso, devemos realizar um **commit**, ou seja, devemos dizer que os arquivos estão prontos para serem enviados ao repositório remoto

```
Tiago Bachiega@Tiago-PC MINGW64 ~/Desktop/GitTest (marcos)
$ git commit -m "Realizadas alterações no Teste1.txt"
[marcos dd0dd90] Realizadas alterações no Teste1.txt
 1 file changed, 1 insertion(+)
```

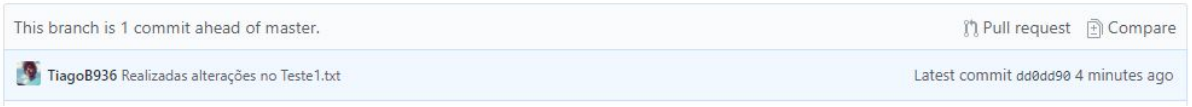
É de extrema importância para o acompanhamento do projeto a inserção de **mensagens simples e objetivas** informando quando é o propósito do commit que está sendo dado. Pode-se perceber que utiliza-se o parâmetro **-m** seguido da **mensagem entre aspas** para isso.

Para efetivamente **enviar** as alterações para o servidor, devemos enviá-lo para a respectiva **branch remota**

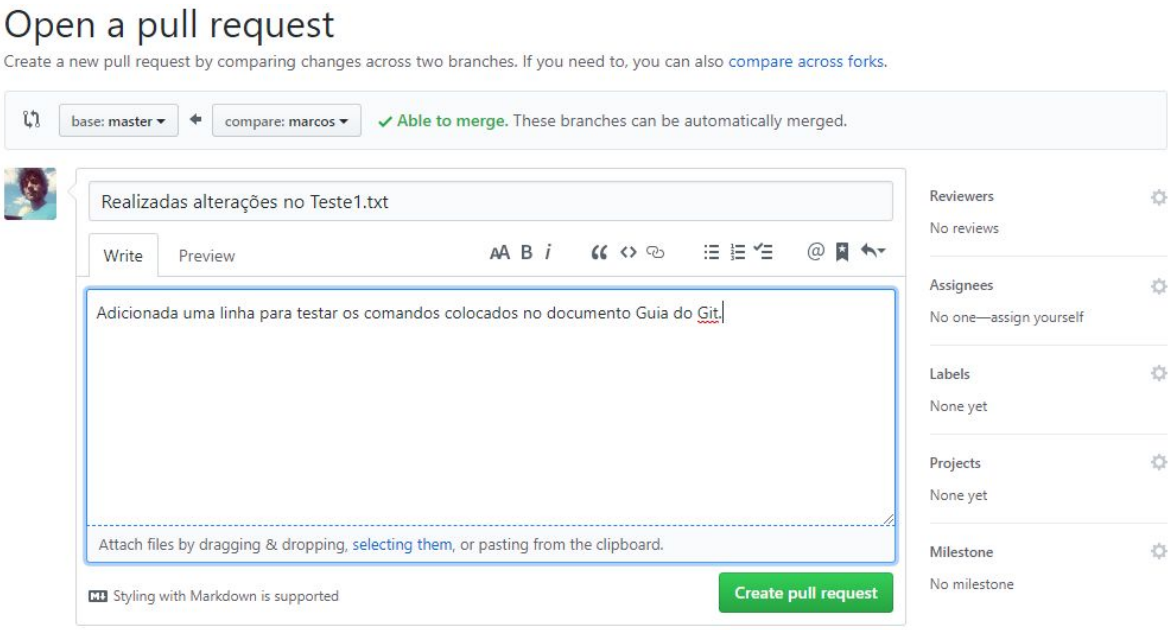
```
Tiago Bachiega@Tiago-PC MINGW64 ~/Desktop/GitTest (marcos)
$ git push origin marcos
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 298 bytes | 74.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/TiagoB936/GitTest.git
   3df49e6..dd0dd90  marcos -> marcos
```

5.4 Exigindo integração com o código fonte

No site do GitHub, devemos acessar nossa branch para ver se o commit realmente está contabilizado. Deve haver uma mensagem como esta se tudo estiver certo:



Para pedir que o **Git Owner**, o gerenciador do Git, inclua as alterações no código fonte, devemos clicar um **Pull Request**. Na tela seguinte, devemos dar um nome sucinto e descritivo para o Pull Request e escrever uma mensagem mais detalhada sobre o que foi alterado. Então, basta clicar em **Create pull request** e esperar o gerenciador aceitar as alterações.



5.6 Git Owner

Git Owner é o nome comumente dado à pessoa responsável por gerenciar o Git. Apesar de todos os membros poderem aceitar os Pull Requests, é muito importante que apenas o Git Owner o faça e que nenhum membro aceite o próprio Pull Request. As funções do Git Owner são:

1. Aceitar ou negar os Pull Requests
2. Solicitar modificações caso necessário
3. Manter o git organizado de forma correta